UNIVERSIDADE FEDERAL DE SÃO CARLOS
Departamento de Computação

Giovanni Castilho Brogiato

Programação linear, linear inteira e os algoritmos Simplex e Branch and Bound: Problemas e aplicações em otimização

São Carlos - SP
2021

Giovanni Castilho Brogiato

Programação linear, linear inteira e os algoritmos Simplex e Branch and Bound: Problemas e aplicações em otimização

Trabalho de graduação apresentado ao programa de graduação em Engenharia de Computação da Universidade Federal de São Carlos, como parte dos requisitos para obtenção do título de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Alexandre Levada

São Carlos - SP
2021

Resumo

Problemas de otimização são aqueles em que se tem como objetivo a maximização ou minimização de uma função, chamada de função objetivo. No processo de otimização são estabelecidas algumas restrições às variáveis de decisão, as quais se puderem assumir valores contínuos configuram um problema de programação linear.
Em um problema geral de programação linear inteira, busca-se minimizar uma função de custo linear sobre todos os vetores n-dimensionais x sujeitos a um conjunto de restrições lineares de igualdade e desigualdade, bem como restrições de integralidade em algumas ou todas as variáveis em x.

$$\begin{aligned} &\min c^T x \\ &\text{s.t } Ax = b \\ &x \geq 0 \\ &x \in Z^n \end{aligned}$$

Se apenas algumas das variáveis $x_i \in x$ são restritas para assumir valores inteiros, e outras podem assumir valores reais, então o problema é chamado de problema de programação linear inteira mista (MILP). Se a função objetivo e/ou as restrições são funções não lineares, o problema é chamado de problema de programação não linear inteira mista (MINLP).
Se todas as variáveis $x_i \in x$ forem restritas para assumir valores inteiros, então o problema é chamado de problema de programação linear inteira puro.
Se todas as variáveis $x_i \in x$ são restritas a assumir valores binários (0 ou 1), então o problema é chamado de problema de otimização binária, que é um caso especial de um problema de programação linear inteira puro.

O método branch and bound não é uma técnica de solução especificamente limitada a problemas de programação de inteiros. É uma abordagem de solução que pode ser aplicada a vários tipos diferentes de problemas. A abordagem de ramificação e limite é baseada no princípio de que o conjunto total de soluções viáveis pode ser dividido em subconjuntos menores de soluções. Esses subconjuntos menores podem então ser avaliados sistematicamente até que a melhor solução seja encontrada.
Palavras-chave: Programação linear inteira. Branch and Bound. Simplex. Problemas de otimização.

Abstract

Optimization problems are those in which the objective is to maximize or minimize a function, called an objective function. In the optimization process, restrictions on decision variables are determined, according to which continuous values can be accepted, which constitute a linear programming problem.
In a general problem of integer linear programming, we seek to minimize a linear cost function over all n-dimensional vectors x subject to a set of linear restrictions of equality and inequality, as well as integrality restrictions in some or all variables in x.

$$\min c^T x$$
$$\text{s.t } Ax = b$$
$$x \geq 0$$
$$x \in Z^n$$

If only some of the variables $x_i \in x$ are restricted to take integer values, and others can take real values, then the problem is called a mixed integer linear programming problem (MILP). If the objective function and/or the constraints are nonlinear functions, the problem is called a mixed integer nonlinear programming problem (MINLP).
If all variables $x_i \in x$ are restricted to assume integer values, then the problem is called a pure integer linear programming problem.
If all variables $xi \in x$ are restricted to assuming binary values (0 or 1), then the problem is called a binary optimization problem, which is a special case of a problem of pure integer linear programming.

The branch and bound method is not a solution technique specifically limited to integer programming problems. It is a solution approach that can be applied to many different types of problems. The branch and boundary approach is based on the principle that the total set of viable solutions can be divided into smaller subsets of solutions. These smaller subsets can then be evaluated systematically until the best solution is found.
Keywords: Integer linear programming. Branch and Bound. Simplex. Optimization problem.

Lista de Ilustrações

Lista de Tabelas

Sumário

1 Introdução

Neste capítulo são apresentadas a contextualização e motivação deste trabalho, bem como os objetivos e a estrutura do texto.

1.1 Contextualização

Um problema de otimização matemática é aquele em que alguma função é maximizada ou minimizada em relação a um determinado conjunto de restrições. A função a ser minimizada ou maximizada é chamada de função objetivo e o conjunto de restrições é chamado de região viável (ou região de restrição). Um LP é um problema de otimização em que a função objetivo é uma função linear, ou seja, apresenta a forma (Hillier; Lieberman; 1967):

$$c_1x_1 + c_2x_2 + \cdots + c_ix$$

com $c_i \in R$ i = 1,...,n, e a região viável sendo um conjunto de soluções que respeitam um número finito de equações e inequações da forma:

$$a_ix_i + a_{i2}x_{i2} + ...+ a_{in}x_{in} \leq b_i \quad i=1,...s$$

e

$$a_ix_i + a_{i2}x_{i2} + ...+ a_{in}x_{in} = b_i \quad i=s+1,...m$$

A programação linear é uma ferramenta extremamente poderosa e tem uma ampla gama de aplicações em problemas de otimização, tais como alocação de recursos, programação de produção, otimização de portfólio e estimativa de parâmetros.
Uma breve história da programação linear:

- Em 1762, Lagrange resolveu problemas de otimização tratáveis com restrições de igualdade simples.
- Em 1820, Gauss resolveu o sistema linear de equações pelo que hoje é chamado de eliminação de Gauss. Em 1866, Wilhelm Jordan refinou o método para encontrar erros de quadrados mínimos como uma medida de adequação. Agora é conhecido como Método Gauss-Jordan.
- Em 1945, surgiu o computador digital.
- Em 1947, Dantzig inventou o Método Simplex.
- Em 1968, Fiacco e McCormick introduziram o Método de Ponto Interior.
- Em 1984, Karmarkar aplicou o Método Interior para resolver Problemas Lineares adicionando sua análise inovadora.

A programação linear provou ser uma ferramenta extremamente poderosa, tanto na modelagem de problemas do mundo real quanto na teoria matemática amplamente aplicável. No entanto, muitos problemas de otimização interessantes

não são lineares. O estudo de tais problemas envolve uma mistura diversa de álgebra linear, cálculo multivariado, análise numérica e técnicas de computação. Áreas importantes incluem o projeto de algoritmos computacionais (incluindo técnicas de pontos interiores para programação linear), a geometria e análise de conjuntos e funções convexas e o estudo de problemas especialmente estruturados, como a programação quadrática. A otimização não linear fornece percepções fundamentais sobre a análise matemática e é amplamente usada em uma variedade de campos, como projeto de engenharia, análise de regressão, controle de estoque, exploração geofísica e economia.

1.2 Motivação

De outubro de 2019 a agosto de 2020 tive a oportunidade de realizar um estágio na Uber, empresa que atua no ramo de corridas de carros compartilhados. Durante todo o estágio participei de um projeto que envolvia otimização e programação linear, e ao buscar por materiais na internet tive dificuldades de encontrar trabalhos com linguagem fácil e voltado a aplicação prática dos conceitos, me motivando a desenvolver a revisão bibliográfica que compõe esse trabalho.

O presente trabalho é de relevância para a comunidade acadêmica, visto que ele dá um panorama sobre programação linear (PL), a programação inteira e os algoritmos branch and bound, Simplex e algumas aplicações reais.

Com exemplo de programação linear, Hillier e Lieberman (2013) nos trazem um exemplo prático da empresa de vidro, onde essa empresa busca atingir o melhor mix da linha de produção com o objetivo de se aumentar o lucro dos produtos feitos pela empresa.

Outro exemplo que os autores trazem é da aplicação em pesquisas de câncer do Memorial Sloan-Kettering Cancer Center (MSKCC). Nesse caso, é mostrado como a programação linear na aplicação de terapia de câncer de próstata. Conforme está no exemplo, em questão de minutos o computador faz a otimização do planejamento de operação de inserção de sementes na próstata do paciente.

Já sobre o algoritmo de branch and bound, Yamashita e Morabito (2007), trata em seu trabalho sobre a aplicação desse algoritmo em problemas de determinação da programação de projetos com custos de disponibilidade de recursos em vários modos de efetivação das atividades.

Esse algoritmo, conforme esses autores, visa minimizar o instante de término da última atividade do projeto, avaliando diversos módulos de execução e respeitando as relações de precedência das atividades e disponibilidade de recursos.

Desse modo, pode-se ver que esses diversos temas são de relevância para o mundo acadêmico, pois encabeça novas oportunidades de pesquisas e abre novos ramos de atuação. Já para o mundo empresarial, pode ser importante o seu estudo objetivando otimização de custos e aumento nas margens operacionais das empresas.

1.3 Objetivos

O presente trabalho tem como principal objetivo servir como material introdutório àqueles que tenham interesse em iniciar os estudos em problemas de otimização e programação linear. Desta forma, o trabalho conta com uma linguagem de fácil acesso e aborda os principais conceitos que permeiam o tema sempre de uma perspectiva prática, sem deixar de lado o entendimento da teoria e da matemática que dão origem aos algoritmos apresentados.
Além disso, objetiva-se trazer à luz da pesquisa acadêmica conceitos apresentados em diversos trabalhos que se correlacionam com o tema exposto e busca realizar uma revisão bibliográfica deles. Além disto, objetiva-se também problematizar sobre a importância da otimização tanto nos meios acadêmicos quanto no meio empresarial.

1.4 Estrutura do Trabalho

O presente estudo está estruturado da seguinte maneira:

1. Introdução: Onde se busca fazer um rápido discurso sobre o tema, apresenta-se seus objetivos e as razões de sua importância;
2. Desenvolvimento: Onde se busca realmente mostrar o tema, seus pormenores e demonstrar sua aplicabilidade;
3. Conclusão: Onde se faz uma rápida síntese do exposto e descreve-se possíveis outros ramos de suporte da pesquisa.

2 O Teorema fundamental da programação linear

Teorema 2.1 (Teorema Fundamental da Programação Linear) (Taha; 2019) *O máximo de uma função objetivo de um problema de programação linear (LPP) pertence a um dos vértices da região viável (P_F), dado que P_F é limitada e não-vazia.*

Prova: Para a maximização de um *LPP:*

Assumindo que o máximo ocorre em algum ponto de P_F , $X_0 \in P_F$ e portanto X_0 é qualquer ponto de P_F e que $X_1, X_2 \dots X_n$ são vértices da região viável.

Partindo da hipótese de que $f(X_0)$ é o valor máximo da função objetivo:

$\Rightarrow f(X_i) \leq f(X_0) \; \forall \; X_i \in P_F$ ... ***(1)***

Uma vez que P_F é convexo, limitado e não vazio, P_F contém um número finito de vértices ($X_1, X_2 \dots X_n$).

$\therefore$ X_0 pode ser escrito como uma combinação linear convexa (clc) de $X_1, X_2 \dots X_n$:

$\Rightarrow X_0 = a_1X_1 + a_2X_2 + \dots + a_nX_n$ s.t $a_i \geq 0$, $\sum ai = 1$

Como f é uma função linear:

$f(X_0) = f(a_1X_1 + a_2X_2 + \dots + a_nX_n) = a_1f(X_1) + a_2f(X_2) + \dots + a_nf(X_n)$ ***(2)***

Fazendo $\max\{f(X_1), f(X_2) \dots, f(X_n)\} = f(X_k)$
$\Rightarrow \; f(X_i) \leq f(X_k)$ para i = 1,2,...,n

À partir de ***(2)***:
$f(X_0) \leq a_1f(X_k) + a_2f(X_k) + \dots + a_nf(X_k) \leq (a_1 + a_2 + \dots a_n)f(X_k)$

$\Rightarrow f(X_0) \leq f(X_k)$ ***(3)***
A equação ***(3)*** contradiz a equação ***(1),*** e portanto $X_0 = X_k$ provando o teorema.

3 Programação linear e o algoritmo Simplex

O método Simplex é uma abordagem para resolver modelos de programação linear manualmente usando variáveis de folga (slack variables), e variáveis de pivô como um meio para encontrar a solução ótima de um problema de otimização (Williams; 1990) . Um programa linear é um método para alcançar o melhor resultado, dada uma equação máxima ou mínima com restrições lineares. A maioria dos problemas lineares pode ser resolvida usando um solucionador online como MatLab, mas o método Simplex é uma técnica para resolver problemas lineares manualmente. Para resolver um modelo de programação linear usando o método Simplex, as seguintes etapas são necessárias:

- Forma padrão
- Introdução de variáveis de folga
- Criação da tabela
- Variáveis pivô
- Criação de uma nova tabela
- Verificação da otimização
- Identificação dos valores ótimos

Etapa 1: Deixando o problema na forma padrão

A forma padrão é o formato base para todos os programas lineares antes de resolvê-lo e tem três requisitos: (1) deve ser um problema de maximização, (2) todas as restrições lineares devem estar em uma desigualdade menor ou igual a , (3) todas as variáveis são não negativas (Gass; 1990) . Esses requisitos podem sempre ser satisfeitos transformando qualquer programa linear dado usando álgebra básica e substituição. A forma padrão é necessária porque cria um ponto de partida ideal para resolver o método Simplex da forma mais eficiente possível, bem como outros métodos de solução de problemas de otimização. Para melhor compreensão do funcionamento do algoritmo, o seguinte problema será levado em consideração:

$$\text{Minimizar: } -z = -8x_1 - 10x_2 - 7x_3$$
$$\text{s.t. : } \quad x_1 + 3x_2 + 2x_3 \leq 10$$
$$-x_1 - 5x_2 - x_3 \geq -8$$
$$x_1, x_2, x_3 \geq 0$$

Para transformar um problema linear de minimização em um problema linear de maximização, basta multiplicar os lados esquerdo e direito da função objetivo por -1:

$$-1 \times (-z = -8x_1 - 10x_2 - 7x_3)$$
$$z = 8x_1 + 10x_2 + 7x_3$$
$$\text{Maximizar: } z = 8x_1 + 10x_2 + 7x_3$$

A transformação de restrições lineares de uma desigualdade “maior ou igual a” uma desigualdade “menor ou igual a” pode ser feita de maneira semelhante ao que foi feito com a função objetivo. Multiplicando por -1 em ambos os lados, a desigualdade pode ser alterada:

$$-1 \times (-x_1 - 5x_2 - x_3 \geq -8)$$
$$x_1 + 5x_2 + x_3 \leq 8$$

Uma vez que o modelo está na forma padrão, as variáveis de folga podem ser adicionadas conforme mostrado na Etapa 2 do método Simplex.

Etapa 2: Introduzindo variáveis de folga

Variáveis de folga são variáveis adicionais que são introduzidas nas restrições lineares de um programa linear para transformá-las de restrições de desigualdade em restrições de igualdade. Se o modelo estiver no formato padrão, as variáveis de folga sempre terão um coeficiente +1. As variáveis de folga são necessárias nas restrições para transformá-las em igualdades solucionáveis com uma resposta definida.

$$x_1 + 3x_2 + 2x_3 + \mathbf{s_1} = 10$$
$$x_1 + 5x_2 + x_3 + \mathbf{s_2} = 8$$
$$x_1, x_2, x_3, \mathbf{s_1}, \mathbf{s_2} \geq 0$$

Depois que as variáveis de folga são introduzidas, a tabela pode ser configurada para verificar a otimização, conforme descrito na Etapa 3.

Etapa 3: Configurando a tabela

Uma tabela Simplex é usada para executar operações de linha no modelo de programação linear, bem como para verificar uma solução para otimização. A tabela consiste no coeficiente correspondente às variáveis de restrição linear e aos coeficientes da função objetivo. No quadro abaixo, a linha superior em negrito da tabela indica o que cada coluna representa. As duas linhas a seguir representam os coeficientes da variável de restrição linear do modelo de programação linear e a última linha representa os coeficientes da variável da função objetivo:

$$\text{Maximizar: } z = 8x_1 + 10x_2 + 7x_3$$
$$x_1 + 3x_2 + 2x_3 + \mathbf{s_1} = 10$$
$$x_1 + 5x_2 + x_3 + \mathbf{s_2} = 8$$

Tabela 1 - Primeira tabela do Simplex

x1	x2	x3	s1	s2	z	b
1	3	2	1	0	0	10
1	5	1	0	1	0	8
-8	-10	-7	0	0	1	0

Assim que a tabela for concluída, o modelo pode ser verificado para uma solução ideal, conforme mostrado na Etapa 4.

Etapa 4: Verificação da otimização

A solução ótima de um modelo de programação linear de maximização são os valores atribuídos às variáveis na função objetivo para fornecer o maior valor de z. A solução ótima existiria nos pontos de canto do gráfico de todo o modelo. Para verificar a otimização usando a tabela, todos os valores na última linha devem conter valores maiores ou iguais a zero. Se um valor for menor que zero, significa que a variável não atingiu seu valor ideal. Como visto na tabela anterior, três valores negativos existem na linha inferior, indicando que essa solução não é a ideal. Se uma tabela não for ideal, a próxima etapa é identificar a variável pivô na qual basear uma nova tabela, conforme descrito na Etapa 5.

Etapa 5: Identificando as variáveis pivô

A variável pivô é usada em operações de linha para identificar qual variável se tornará o valor da unidade. A variável pivô pode ser identificada observando a linha inferior do quadro e o indicador. Supondo que a solução não seja a ideal, o menor valor negativo na linha inferior deve ser escolhido. Um dos valores que se encontram na coluna deste valor será a variável pivô. Para encontrar o indicador, é preciso dividir os valores beta das restrições lineares por seus valores correspondentes da coluna que contém a possível variável dinâmica. A interseção da linha com o menor indicador não negativo e o menor valor negativo na linha inferior se tornará a variável pivô.
No exemplo mostrado abaixo, -10 é o menor negativo na última linha. Isso designará a coluna x2 para conter a variável pivô. A resolução para o indicador nos dá um valor de 10/3 para a primeira restrição e um valor de 8/5 para a segunda restrição. Como 8/5 é o menor indicador não negativo, o valor do pivô estará na segunda linha e terá o valor 5. Agora que a nova variável pivô foi identificada, o novo quadro pode ser criado na Etapa 6 para otimizar a variável e encontrar a nova solução ótima possível.

Tabela 2 - Segunda tabela do Simplex

x1	x2	x3	s1	s2	z	b
1	3	2	1	0	0	10
1	**5**	1	0	1	0	8
-8	-10	-7	0	0	1	0

Etapa 6: Criação de uma nova tabela

O novo quadro será usado para identificar uma nova solução ótima possível. Agora que a variável dinâmica foi identificada na Etapa 5, as operações de linha podem ser realizadas para otimizar a variável dinâmica enquanto mantém o restante do quadro equivalente.

1) Para otimizar a variável dinâmica, ela precisará ser transformada em um valor unitário (valor 1). Para transformar o valor, é preciso multiplicar a linha que contém a variável pivô pelo recíproco do valor pivô. No exemplo abaixo, a variável dinâmica é originalmente 5, portanto, é preciso multiplicar a linha inteira por 1/5.

Tabela 3 - Terceira tabela do Simplex

x1	x2	x3	s1	s2	z	b
1/5	**1**	1/5	0	1/5	0	8/5

2) Após a determinação do valor da unidade, os outros valores na coluna que contém o valor da unidade se tornarão zero. Isso ocorre porque o x2 na segunda restrição está sendo otimizado, o que exige que x2 nas outras equações seja zero.

Tabela 4 - Quarta tabela do Simplex

x1	x2	x3	s1	s2	z	b
	0					
1/5	**1**	1/5	0	1/5	0	8/5
	0					

3) Para manter a tabela equivalente, as outras variáveis não contidas na coluna ou linha da pivô devem ser calculadas usando os novos valores da pivô. Para cada novo valor, é preciso multiplicar o negativo do valor na coluna pivô antiga pelo valor na nova linha pivô que corresponde ao valor que está sendo calculado. Em seguida, adicionar isso ao valor antigo do quadro antigo para produzir o novo valor para o novo quadro. Esta etapa pode ser resumida na equação seguinte:

Novo valor da tabela = (valor negativo na coluna pivô da tabela antiga) x (valor na linha pivô da nova tabela) + (valor antigo da tabela)

Tabela antiga:

Tabela 5 - Quinta tabela do Simplex

x1	x2	x3	s1	s2	z	b
1	3	2	1	0	0	10
1	**5**	1	0	1	0	8
-8	-10	-7	0	0	1	0

Nova tabela:

Tabela 6 - Sexta tabela do Simplex

x1	x2	x3	s1	s2	z	b
2/5	0	7/5	1	-3/5	0	26/5
1/5	**1**	1/5	0	1/5	0	8/5
-6	0	-5	0	2	1	16

Exemplos numéricos são fornecidos abaixo para ajudar a explicar este conceito um pouco melhor.

Exemplos numéricos:

I. Para encontrar o valor s2 na linha 1:
Novo valor da tabela = (valor negativo na coluna pivô da tabela antiga) * (valor na linha pivô da nova tabela) + (valor antigo da tabela)
Novo valor da tabela = (-3) *($\frac{1}{5}$) + 0 = -$\frac{3}{5}$

II. Para encontrar o valor x1 na linha 3:

Novo valor da tabela = (valor negativo na coluna pivô da tabela antiga) * (valor na linha pivô da nova tabela) + (valor antigo da tabela)

Novo valor da tabela = (10) *($\frac{1}{5}$) - 8 = -6

Assim que o novo quadro for concluído, o modelo pode ser verificado para uma solução ideal.

Etapa 7: Criação de uma nova tabela

Conforme explicado na Etapa 4, a solução ótima de um modelo de programação linear de maximização são os valores atribuídos às variáveis na função objetivo para fornecer o maior valor de Z. A otimização precisará ser verificada após cada nova tabela para ver se uma nova variável pivô precisa ser identificada. Uma solução é considerada ótima se todos os valores na linha inferior forem maiores ou iguais a zero e assim as etapas 8 a 11 podem ser ignoradas. Se houver valores negativos, a solução ainda não é a ideal e um novo ponto pivô precisará ser determinado, o que é demonstrado na Etapa 8.

Etapa 8: Identificando uma nova variável

Se a solução foi identificada como não ótima, uma nova variável pivô precisará ser determinada. Essa variável foi introduzida na Etapa 5 e é usada em operações de linha para identificar qual variável se tornará o valor da unidade. A variável pivô pode ser identificada pela interseção da linha com o menor indicador não negativo e o menor valor negativo na linha inferior.

Tabela 7 - Sétima tabela do Simplex

x1	x2	x3	s1	s2	z	b
2/5	0	7/5	1	-3/5	0	26/5
1/5	1	1	0	1/5	0	8/5
-6	0	-5	0	2	1	0

Etapa 9: Criando uma nova tabela

Depois que a nova variável pivô for identificada, uma nova tabela precisará ser criada:

I. É preciso transformar a variável dinâmica em 1 multiplicando a linha que a contém pelo recíproco de seu valor. No quadro abaixo, o valor do pivô era 1/5, então tudo é multiplicado por 5.

Tabela 8 - Oitava tabela do Simplex

x1	x2	x3	s1	s2	z	b
1	5	1	0	1	0	8

II. Em seguida, é necessário fazer com que os outros valores na coluna da variável pivô sejam zero. Isso é feito pegando negativo do valor antigo na coluna pivô e multiplicando-o pelo novo valor na linha pivô. Esse valor é então adicionado ao valor antigo que está sendo substituído.

Tabela 9 - Nona tabela do Simplex

x1	x2	x3	s1	s2	z	b
0	-2	1	1	-1	0	2
1	5	1	0	1	0	8
0	30	1	0	8	1	64

Etapa 10: Verificação da otimização

Usando o novo quadro, verifica-se se a solução ótima foi atingida. Explicado na Etapa 4, uma solução ótima aparece quando todos os valores na linha inferior são maiores ou iguais a zero. Se ainda existirem valores negativos, é preciso repetir as etapas 8 e 9 até que uma solução ótima seja obtida.

Etapa 11: Verificação da otimização

Assim que a tabela for comprovada como ótima, os valores ideais podem ser identificados. Estes podem ser encontrados distinguindo as variáveis básicas e não básicas. Uma variável básica pode ser classificada por ter um único valor 1 em sua coluna e o resto ser todos zeros. Se uma variável não atender a esses critérios, ela será considerada não básica. Se uma variável não é básica, significa que a solução ótima dessa variável é zero. Se uma variável for básica, a linha que contém o valor 1 corresponderá ao valor beta. O valor beta representará a solução ótima para a variável fornecida.

Tabela 10 - Décima tabela do Simplex

x1	**x2**	**x3**	**s1**	**s2**	**z**	**b**
0	-2	1	1	-1	0	2
1	5	1	0	1	0	8
0	30	1	0	8	1	64

Variáveis básicas: x1, s1, z
Variávei não básicas: x2, x3, s2

Para a variável x1, o 1 é encontrado na segunda linha. Isso mostra que o valor x1 ideal é encontrado na segunda linha dos valores beta, que é 8.
A variável s1 tem um valor 1 na primeira linha, mostrando que o valor ideal é 2 na coluna beta. Devido ao fato de s1 ser uma variável de folga, ela não está realmente incluída na solução ótima, pois a variável não está contida na função objetivo.
A variável zeta possui um 1 na última linha. Isso mostra que o valor objetivo máximo será 64 da coluna beta.
A solução final tem para cada variável:

Tabela 11 - Valor final de cada variável

$x_1 = 8$	s1 = 2
$x_2 = 0$	s2 = 0
$x_3 = 0$	z = 64

O valor máximo ótimo é 64 e é encontrado em (8,0,0) da função objetivo.

4 O algoritmo branch and bound

Em alguns problemas reais de otimização, surge a necessidade de limitar as variáveis de decisão ao conjunto dos inteiros, surgindo uma nova classe de problemas os quais podem ser resolvidos com a programação linear inteira.
O método branch and bound não é uma técnica de solução especificamente limitada a problemas de programação de inteiros. É uma abordagem de solução que pode ser aplicada a vários tipos diferentes de problemas. A abordagem branch and bound é baseada no princípio de que o conjunto total de soluções viáveis pode ser particionado em subconjuntos menores de soluções. Esses subconjuntos menores podem então ser avaliados sistematicamente até que a melhor solução seja encontrada. Quando essa abordagem é aplicada a um problema de programação inteira, ela é usada em conjunto com a abordagem de solução normal não inteira. Para uma melhor compreensão do método, um problema será resolvido a seguir como exemplo de sua funcionalidade (Hillier; Lieberman; 1967) .
O proprietário de uma oficina mecânica está planejando expandir com a compra de algumas novas máquinas - prensas e tornos. O proprietário estimou que cada prensa comprada aumentará o lucro em $100,00 por dia e cada torno aumentará o lucro em $150,00 por dia. O número de máquinas que o proprietário pode comprar é limitado pelo custo das máquinas e pelo espaço disponível na oficina. Os preços de compra da máquina e os requisitos de espaço são os seguintes.

Tabela 12 - Especificações do Problema

Máquina	**Espaço necessário (m^2)**	**Preço**
Prensa	15	R$8.000
Torno	30	R$4.000

O proprietário tem um orçamento de R$40.000 para a compra de máquinas e 200 metros quadrados de espaço disponível. O proprietário quer saber quantas unidades de cada tipo de máquina deve comprar para maximizar o aumento diário do lucro.
O modelo de programação linear para um problema de programação inteira é formulado exatamente da mesma maneira que o exemplo de programação linear mostrado no capítulo 3 que foi solucionado utilizando o algoritmo simplex. A única diferença é que, neste problema, as variáveis de decisão são restritas a valores inteiros porque o proprietário não pode comprar uma fração, ou parte, de uma máquina. O problema pode ser modelado da seguinte maneira:

$$\text{Maximizar: } Z = 100x_1+150x_2$$
$$\text{s.t. : } \quad 8.000x_1 +4000x_2 \leq 40.000$$
$$15x_1 +30x_2 \leq 200$$

$$x_1, x_2 \geq 0 \in Z$$

onde:

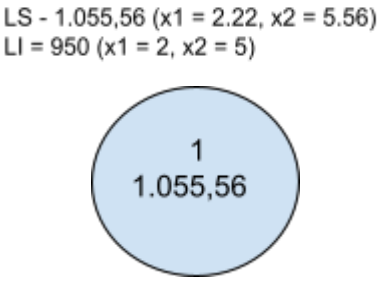
x_1=número de prensas
x_2=número de tornos

As variáveis de decisão neste modelo são restritas a máquinas inteiras. O fato de que ambas as variáveis de decisão, x_1 e x_2, podem assumir qualquer valor inteiro maior ou igual a zero é o que dá a este modelo sua designação como um modelo inteiro total.
Começamos o método branch and bound resolvendo primeiro o problema como um modelo de programação linear regular sem restrições de inteiros (ou seja, as restrições de inteiros são relaxadas). A solução relaxada ideal é:

$$x_1 = 2.22, x_2 = 5.56, Z = 1.055,56$$

O método branch and bound emprega um diagrama que consiste em nós e ramificações como uma estrutura para o processo de solução. O primeiro nó do diagrama de ramificação e limite, mostrado na Figura 4.1, contém a solução de programação linear relaxada mostrada anteriormente e a solução arredondada para baixo.

Figura 4.1 - Primeiro nó do branch and bound



Fonte: Elaborado pelo autor

É importante notar que este nó tem dois limites designados: um limite superior (LS) de R$1.055,56 e um limite inferior (LI) de R$950. O limite inferior é o valor Z para a solução arredondada para baixo, x_1=2 e x_2=5; o limite superior é o valor Z para a solução relaxada, x_1 = 2,22 e x_2 = 5,56. A solução de número inteiro ideal estará entre esses dois limites.
O arredondamento para baixo pode resultar em uma solução abaixo do ideal. Em outras palavras, espera-se um valor Z maior que R$950 seja possível. Não existe a preocupação de que um valor inferior a R$950 possa estar disponível. Portanto, R$950 representa um limite inferior para a solução. Alternativamente, uma vez que Z = $1.055,56 reflete um ponto de solução ótimo no limite do espaço de solução, um valor Z maior não pode ser atingido. Portanto, Z = R$1.055,56 é o limite superior da solução.

Agora que as possíveis soluções viáveis foram reduzidas a valores entre os limites superior e inferior, deve-se testar as soluções dentro desses limites para determinar o melhor. A primeira etapa no método branch and bound é criar dois subconjuntos de solução a partir da solução atual relaxada. Isso é feito observando o valor da solução relaxada para cada variável:

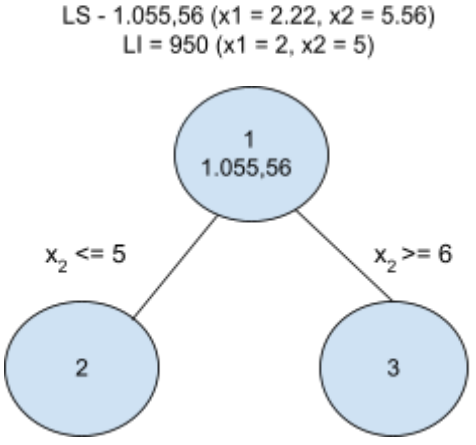
$$x_1 = 2.22$$
$$x_2 = 5.56$$

e ver qual deles está mais longe do valor inteiro arredondado para baixo (ou seja, qual variável tem a maior parte fracionária). A porção 0,56 de 5,56 é a maior parte fracionária; assim, x_2 será a variável que será “ramificada”.
Como x_2 deve ser um valor inteiro na solução ótima, as seguintes restrições podem ser desenvolvidas.

$$x_2 \leq 5$$
$$x_2 \geq 6$$

Em outras palavras, x_2 pode ser 0, 1, 2, 3, 4, 5 ou 6, 7, 8..., mas não pode ser um valor entre 5 e 6, como 5,56. Essas duas novas restrições representam os dois subconjuntos de solução para a abordagem de solução. Cada uma dessas restrições será adicionada ao modelo de programação linear, que então será resolvido normalmente para determinar uma solução relaxada. Esta sequência de eventos é mostrada no diagrama de ramificação e limite na Figura 4.2. As soluções nos nós 2 e 3 serão as soluções relaxadas obtidas resolvendo o modelo de exemplo com o restrições apropriadas adicionadas.

Figura 4.2 - Subset da variável x_2



Fonte: Elaborado pelo autor

A solução no nó 2 é encontrada a partir do seguinte modelo com a restrição $x_2 \leq 5$ adicionada:

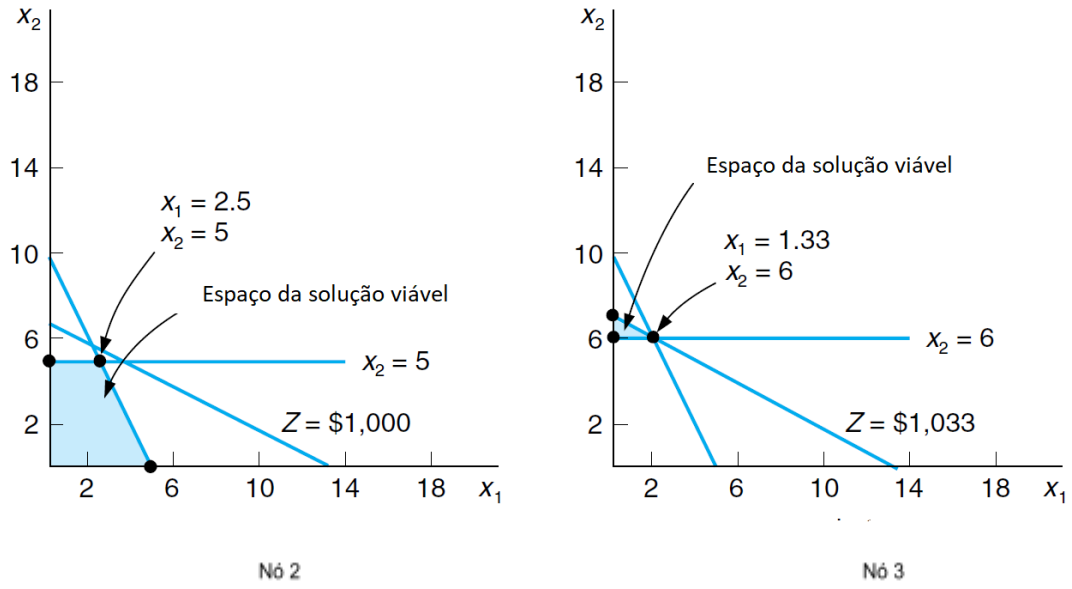
$$\text{Maximizar: } Z = 100x_1+150x_2$$
$$\text{s.t. : } \quad 8.000x_1 +4000x_2 \leq 40.000$$
$$15x_1 +30x_2 \leq 200$$
$$x_2 \leq 5$$
$$x_1, x_2 \geq 0$$

A solução ótima deste modelo com a restrição de inteiros relaxada (utilizando o computador) é $x_1 = 2.5$, $x_2 = 5$, $Z = 1000$.
Em seguida, a solução no nó 3 é encontrada através do modelo adicionando a restrição $x_2 \geq 6$:

$$\text{Maximizar: } Z = 100x_1+150x_2$$
$$\text{s.t. : } \quad 8.000x_1 +4000x_2 \leq 40.000$$
$$15x_1 +30x_2 \leq 200$$
$$x_2 \geq 6$$
$$x_1, x_2 \geq 0$$

A solução ótima deste modelo com a restrição de inteiros relaxada (utilizando o computador) é $x_1 = 1.33$, $x_2 = 6$, $Z = 1033,33$.

Figura 4.3 - Espaço viável de soluções dos nós 2 e 3

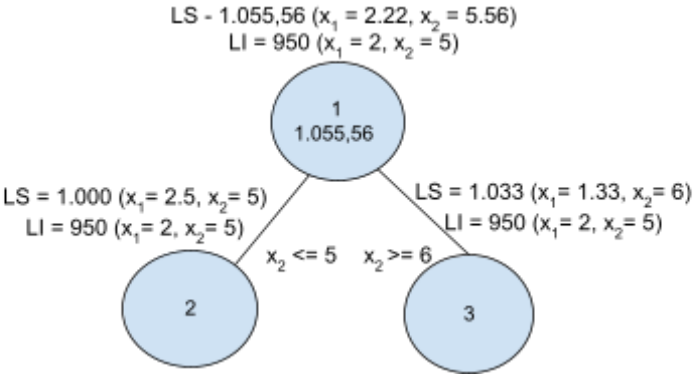


Fonte: Elaborado pelo autor

É importante notar que para o nó 2 da figura 5.3, a solução $x_1 = 2.5$ e $x_2 = 5$ resulta em um valor máximo de Z = $1.000, o qual configura o limite superior para esse nó. Em relação ao nó 3, é possível notar que a solução $x_1 = 1.33$ e $x_2 = 6$ resulta em um valor máximo de Z = $1.033. Portanto 1.033 é o limite superior para o nó 3. O limite inferior para ambos os nós é a solução inteira máxima, uma vez que nenhuma das

soluções é totalmente inteira, o limite inferior permanece $950, a solução inteira já obtida no primeiro nó arredondando a solução relaxada para baixo. A figura 4.4 mostra a adição do limite superior e inferior para cada nó.

Figura 4.4 - Nós 2 e 3 do branch and bound



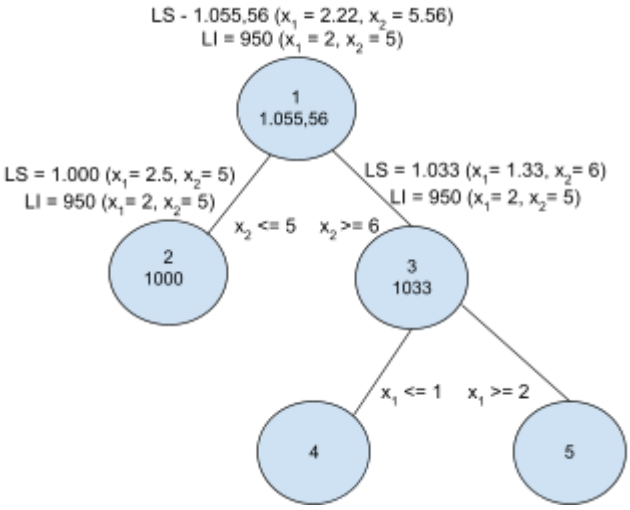
Fonte: Elaborado pelo autor

Uma vez que ainda não há solução viável inteira e ótima, é preciso continuar a ramificação do modelo a partir dos nós 2 ou 3. Como o valor da solução possível no nó 3 é maior, ele será o escolhido para a próxima ramificação.
Agora os passos seguidos anteriormente para o nó 1 serão repetidos no nó 3. Primeiramente a variável com a maior parte fracionária é selecionada. Como x_2 é um valor inteiro e $x_1 = 1.33$, x_1 será a variável selecionada, da qual duas restrições podem ser estabelecidas:

$$x_1 \leq 1$$
$$x_1 \geq 2$$

Dando origem ao diagrama representado na figura 4.5:

Figura 4.5 - Subset de soluções para x_1 no nó 3



Fonte: Elaborado pelo autor

O modelo de programação linear com as novas restrições adicionadas deve ser resolvido nos nós 4 e 5. É importante ressaltar, que o modelo não é mais o original e sim com a restrição anteriormente adicionada: $x_2 \geq 6$. Considerando o nó 4 primeiro:

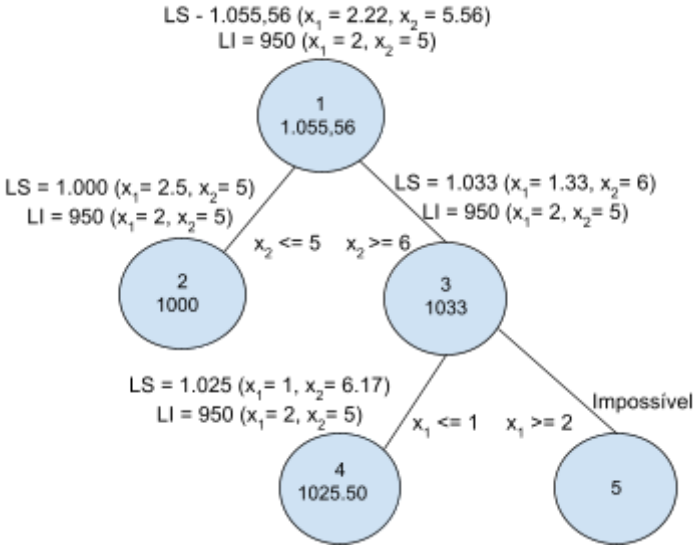
$$\text{Maximizar: } Z = 100x_1+150x_2$$
$$\text{s.t. : } \quad 8.000x_1 +4000x_2 \leq 40.000$$
$$15x_1 +30x_2 \leq 200$$
$$x_2 \geq 6$$
$$x_1 \leq 1$$
$$x_1, x_2 \geq 0$$

A solução ótima deste modelo com a restrição de inteiros relaxada (utilizando o computador) é $x_1 = 1$, $x_2 = 6.17$, $Z = 1025$.
Para o nó 5:

$$\text{Maximizar: } Z = 100x_1+150x_2$$
$$\text{s.t. : } \quad 8.000x_1 +4000x_2 \leq 40.000$$
$$15x_1 +30x_2 \leq 200$$
$$x_2 \geq 6$$
$$x_1 \geq 2$$
$$x_1, x_2 \geq 0$$

Não há solução viável para esse modelo, dessa forma não há solução no nó 5, e apenas o nó 4 deve ser avaliado. O diagrama refletindo os resultados é representado na figura 4.6.

Figura 4.6 - Nós 4 e 5 do branch and bound



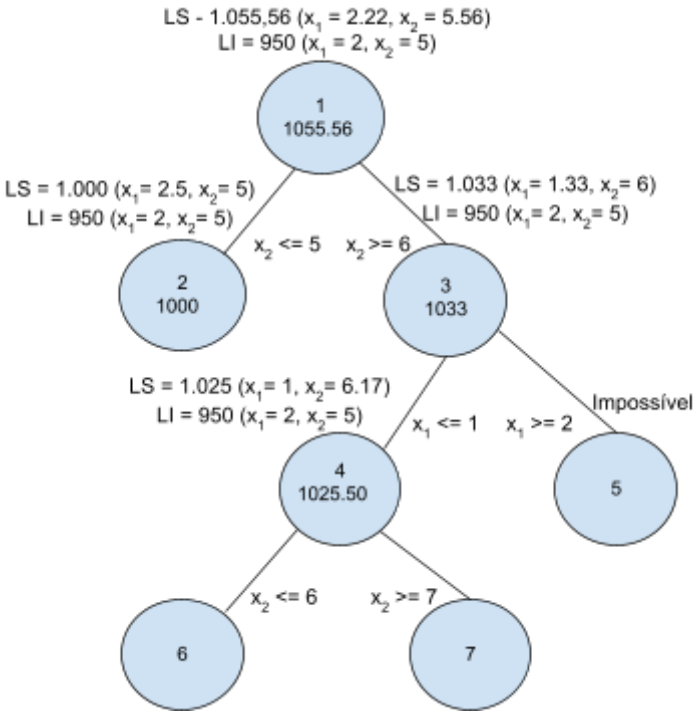
Fonte: Elaborado pelo autor

O diagrama anterior mostra que uma solução ótima inteira ainda não foi alcançada. Uma vez que não há solução partindo do nó 5, não há comparação entre limites superiores dos nós 4 e 5. Comparando os nós 2 e 4, deve-se ramificar o nó 4, uma vez que apresenta um limite superior maior. Como x_1 é um valor inteiro e $x_2 = 6.17$, x_2 será a variável selecionada, da qual duas restrições podem ser estabelecidas:

$$x_2 \leq 6$$
$$x_2 \geq 7$$

O modelo de programação linear com as novas restrições adicionadas deve ser resolvido nos nós 6 e 7. A figura 4.7 mostra o novo diagrama.

Figura 4.7 - Subset de soluções para x_2 no nó 4



Fonte: Elaborado pelo autor

Considerando primeiramente o nó 6:

$$\text{Maximizar: } Z = 100x_1 + 150x_2$$
$$\text{s.t. : } \quad 8.000x_1 + 4000x_2 \leq 40.000$$
$$15x_1 + 30x_2 \leq 200$$
$$x_2 \geq 6$$
$$x_1 \leq 1$$
$$x_2 \leq 6$$
$$x_1, x_2 \geq 0$$

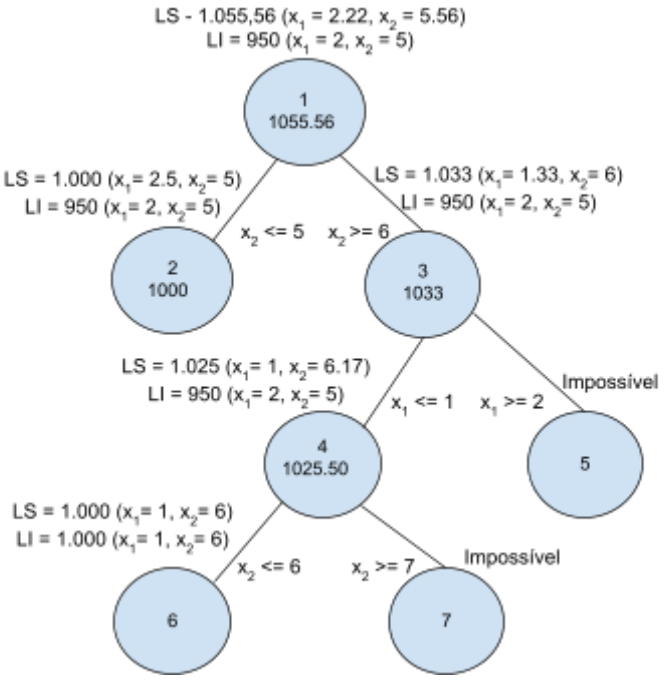
A solução ótima deste modelo com a restrição de inteiros relaxada (utilizando o computador) é $x_1 = 1$, $x_2 = 6$, $Z = 1000$.

Para o nó 7:

$$\text{Maximizar: } Z = 100x_1+150x_2$$
$$\text{s.t. : } \quad 8.000x_1 +4000x_2 \leq 40.000$$
$$15x_1 +30x_2 \leq 200$$
$$x_2 \geq 6$$
$$x_1 \leq 1$$
$$x_2 \geq 7$$
$$x_1, x_2 \geq 0$$

Para o nó 7 não há solução viável. O diagrama mostrado na figura 4.8 reflete os resultados obtidos. Essa versão do diagrama do algoritmo branch and bound indica a solução inteira ótima para o problema $x_1 = 1$, $x_2 = 6$ a qual foi atingida no nó 6. 1000 é o limite superior, ou seja, o maior valor inteiro que pode ser obtido. Além disso, também é o limite inferior, por ser a solução máxima inteira alcançada nesse ponto.

Figura 4.8 - Diagrama do branch and bound com a solução ótima no nó 6



Fonte: Elaborado pelo autor

Uma comparação do nó 6 com os nós 2, 5 e 7 mostra que uma solução melhor não é possível. O limite superior no nó 2 é 1.000, que é o mesmo obtido no nó 6, porém não é possível obter uma solução melhor a partir dele. Para os nós 5 e 7 não há solução viável, e portanto a ramificação a partir deles resultará apenas em soluções inviáveis.

Em geral, a solução ótima inteira é alcançada quando uma solução viável e inteira é encontrada em um nó e o limite superior neste nó é maior ou igual ao limite superior de todos os outros nós finais (nó no final de um ramo).

No contexto do problema original, esta solução indica que se o proprietário da oficina comprar 1 prensa e 6 tornos, terá um aumento no lucro diário de R$1.000,00.
As etapas do algoritmo branch and bound para determinar uma solução ótima e inteira para um problema de maximização (com restrições $\leq$) podem ser resumidas como se segue:

1. Encontrar a solução ótima para o modelo de programação linear com as restrições de inteiro relaxadas.
2. No nó 1, deixar a solução relaxada ser o limite superior e a solução inteira arredondada para baixo o limite inferior.
3. Selecionar a variável com a maior parte fracionária para ramificação. Criar duas novas restrições para esta variável refletindo os valores inteiros particionados. O resultado será uma nova restrição $\leq$ e uma nova restrição $\geq$
4. Criar dois novos nós, um para a restrição $\geq$ e outro para a restrição $\leq$
5. Resolver o modelo de programação linear relaxado com a nova restrição adicionada em cada um desses nós.
6. A solução relaxada é o limite superior em cada nó, e a solução de número inteiro máximo existente (em qualquer nó) é o limite inferior.
7. Se o processo produzir uma solução inteira viável com o maior valor de limite superior de qualquer nó final, a solução inteira ótima foi alcançada. Se uma solução inteira viável não surgir, ramifique do nó com o maior limite superior.
8. Voltar para a etapa 3.

5 Resolvendo problemas de otimização utilizando Python

5.1 A linguagem Python

Python é incrivelmente fácil de usar e aprender para iniciantes e novatos, sendo uma das linguagens de programação mais acessíveis disponíveis, pois possui sintaxe simplificada, o que dá mais ênfase à linguagem natural. Desta forma, os códigos Python podem ser facilmente escritos e executados muito mais rápido do que outras linguagens de programação.
Um ponto importante sobre a versatilidade da linguagem python é que ela pode ser usada em uma série de ambientes, como aplicativos móveis, aplicativos de desktop, desenvolvimento web, programação de hardware e muitos mais. A versatilidade da linguagem a torna mais atraente para uso devido ao seu alto número de aplicações.
Devido ao seu patrocínio corporativo e grande comunidade de suporte, Python tem excelentes bibliotecas que podem ser usadas para selecionar e economizar tempo e esforço no ciclo inicial de desenvolvimento. Existem também muitos serviços de mídia em nuvem que oferecem suporte a várias plataformas por meio de ferramentas semelhantes a bibliotecas, o que pode ser extremamente benéfico.
Bibliotecas com foco específico também estão disponíveis, como *Nltk* para processamento de linguagem natural, *Scikit-learn* para aplicativos de aprendizado de máquina e *PuLP* utilizada para resolver problemas de otimização, a qual foi a escolhida para a exemplificação neste trabalho.

5.2 A biblioteca *PuLP*

PuLP é uma biblioteca para a linguagem Python que permite aos usuários descrever programas matemáticos. A biblioteca funciona inteiramente dentro da sintaxe e expressões idiomáticas naturais da linguagem Python, fornecendo objetos que representam problemas de otimização e variáveis de decisão, e permitindo que restrições sejam estabelecidas de uma forma muito semelhante à expressão matemática original. Para manter a sintaxe o mais simples e intuitiva possível, *PuLP* focou no suporte a modelos lineares e inteiros mistos. *PuLP* pode ser facilmente implementada em qualquer sistema que tenha um interpretador Python, uma vez que não depende de nenhum outro pacote de software. A biblioteca oferece suporte a uma ampla gama de produtos comerciais e de código aberto e pode ser facilmente estendida para oferecer suporte a solucionadores adicionais.

5.3 Resolvendo um exemplo com *PuLP*

O problema trazido no capítulo 4 deste trabalho será utilizado como exemplo de utilização da biblioteca *PuLP.* Desta forma partimos da modelagem inicial do problema:

$$\text{Maximizar: } Z = 100x+150y$$
$$\text{s.t. : } \quad 8.000x + 4000y \leq 40.000$$
$$15x + 30y \leq 200$$
$$x, y \geq 0 \in Z$$

onde:

x=número de prensas
y=número de tornos

O primeiro passo é importar a biblioteca e as respectivas classes que utilizaremos para a solução do modelo, para isso é preciso a criação de um arquivo .py e em seguida a adição do seguinte código:

Python

```
#Importação da biblioteca
from pulp import LpMaximixe, LpProblem, LpStatus, LpVariable
```

Em seguida é preciso inicializar uma instância do problema de programação linear para representar o modelo o qual deve ser solucionado, para isso a biblioteca disponibiliza a classe *LpProblem:*

Python

```
#Criação do modelo
model = LpProblem(name="oficina", sense=LpMaximize)
```

O parâmetro *name* é utilizado para nomear o modelo, enquanto o parâmetro sense é utilizado para escolher se será performado uma minimização (*LpMinimize* ou 1, que é o padrão) ou uma maximização (*LpMaximize* ou -1).
Uma vez que o problema foi definido, deve-se definir as variáveis de decisão como instâncias da classe *LpVariable*:

Python

```
#Inicializar as variáveis de decisão
x = LpVariable(name="x", lowBound=0, cat="Integer" )
y = LpVariable(name="y", lowBound=0, cat="Integer")
```

É necessário informar um limite inferior *lowBound=0* porque o valor padrão é infinito negativo. O parâmetro *upBound* define um limite superior, mas pode ser omitido uma vez que por padrão tem valor de infinito positivo.
Como se trata de um problema de programação linear inteira, é preciso definir o tipo da variável como inteiro utilizando *cat="Integer".* Caso se tratasse de um problema de variáveis contínuas, o valor "Continuos" deveria ter sido utilizado.
Com as variáveis de decisão já definidas, agora é preciso inserir as restrições ao modelo e para isso a seguinte sintaxe é utilizada.

Python

```
#Adicionar restrições ao modelo
model += (8000* x + 4000*y <= 40000, "orçamento")
model += (15* x + 30*y <= 200, "área")
```

No código acima são definidas tuplas as quais armazenam as restrições e seus respectivos nomes, sendo o primeiro elemento uma instância da classe *LpConstraint*.
A função objetivo também pode ser definida de uma forma bastante simples:

Python

```
#Definindo a função objetivo
model += 100* x + 150*y
```

Por fim, o modelo pode ser resolvido através da função *.solve()* do objeto *model*:

Python

```
#Resolvendo o modelo
status = model.solve()
```

Para obter os resultados da otimização o método *.value()* retorna os valores dos atributos, *model.objective* contém o valor da função objetivo enquanto *model.variables()* o valor das variáveis de decisão.

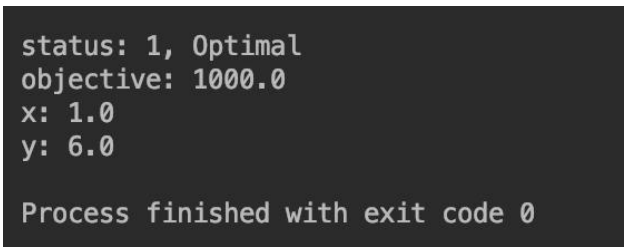
```python
print(f"status: {model.status}, {LpStatus[model.status]}")

print(f"objective: {model.objective.value()}")

for var in model.variables():
    print(f"{var.name}: {var.value()}")
```

Após as linhas acima serem executadas, o seguinte resultado deve aparecer no ambiente de desenvolvimento:

Figura 5.1 - Resultado do modelo de otimização



Fonte: Elaborado pelo autor

No contexto do problema original, esta solução indica que se o proprietário da oficina comprar 1 prensa e 6 tornos, terá um aumento no lucro diário de R$1.000,00.

6 Conclusões

A partir do exposto no trabalho é possível concluir que a programação linear é uma técnica de otimização fundamental usada há décadas em diversos campos da ciência, sendo precisa, relativamente rápida e adequada para uma variedade de aplicações práticas. O domínio das variáveis de decisão em um modelo determina se trata de programação linear contínua, para variáveis contínuas ou programação linear inteira, para variáveis inteiras.

O método simplex é um dos algoritmos mais úteis e eficientes já inventados e ainda é o método padrão empregado em computadores para resolver problemas de otimização. Primeiro, o método assume que um ponto extremo é conhecido. (Se nenhum ponto extremo for fornecido, uma variante do método simplex, é usada para encontrar um ou para determinar que não há soluções viáveis.) Em seguida, usando uma especificação algébrica do problema, um teste determina se o ponto extremo é o ideal. Se o teste de otimização não for aprovado, um ponto extremo adjacente é procurado ao longo de uma aresta na direção para a qual o valor da função objetivo aumenta na taxa mais rápida. Às vezes, pode-se mover ao longo de uma borda e fazer o valor da função objetivo aumentar sem limites. Se isso ocorrer, o procedimento termina com a prescrição da borda ao longo da qual a objetiva vai para o infinito positivo. Caso contrário, um novo ponto extremo é alcançado tendo um valor de função objetivo pelo menos tão alto quanto seu predecessor. A sequência descrita é então repetida. A rescisão ocorre quando um ponto extremo ideal é encontrado ou ocorre o caso ilimitado. Embora, em princípio, as etapas necessárias possam crescer exponencialmente com o número de pontos extremos, na prática o método normalmente converge para a solução ótima em uma série de etapas que é apenas um pequeno múltiplo do número de pontos extremos.

Para problemas de programação linear inteira, um dos algoritmos mais utilizados é o Branch and Bound. Algoritmos de ramificação e limite são usados para encontrar a solução ótima para problemas de otimização matemática combinatória, discreta e geral. Em geral, um algoritmo branch and bound explora todo o espaço de busca de soluções possíveis e fornece uma solução ótima, para isso ocorre a enumeração passo a passo de possíveis soluções candidatas, explorando todo o espaço de pesquisa.

7 Referências

A. Geoffrion and R. Marsten, Integer Programming Algorithms: A Framework and State-of-the-Art Survey, Management Science 18, 1972.

C. Barnhart, E. Johnson, G. Nemhauser, M. Savelsbergh, and P. Vance, Branch-and-Price: Column Generation for Solving Huge Integer Programs, Operations Research 46, 1998.

R. Gomory, Outline of an Algorithm for Integer Solutions to Linear Programs, Bulletin of the American Mathematical Society 64, 1958.

Hillier, Frederick S., and Gerald J. Lieberman: Introduction to operations research, São Francisco, CA, 1967.

Gass, S.: An Illustrated Guide to Linear Programming, Dover Publications, Nova Iorque, 1990.

Williams, H. P.: Model Building in Mathematical Programming, 3d ed., Wiley, Nova Iorque, 1990.

Dantzig, G.B., and M.N. Thapa: Linear Programming 1: Introduction, Springer, Nova Iorque, 1997.

Vanderbei, R. J.: Linear Programming: Foundations and Extensions, Kluwer Academic Publishers, Boston, MA, 1996.

Schriver, A.: Theory of Linear and Integer Programming, Wiley, Nova Iorque, 1986.

Hoffman, K. L., and M. Padberg: “Improving LP-Representations of Zero-One Linear Programs for Branch-and-Cut,” ORSA Journal on Computing, 1991.

Taha, H.A.: Operations research An Introduction, Pearson, 2019.

Mitchell S., O’Sullivan M., Dunning I.: PuLP: A Linear Programming Toolkit for Python, 2011. Disponível em:
<http://www.optimization-online.org/DB_FILE/2011/09/3178.pdf>

Documentação da biblioteca *Pulp*. Disponível em:
<https://projects.coin-or.org/PuLP/export/353/trunk/doc/pulp.pdf>