Análise de Algoritmos de Otimização para Produção de Cerveja

Fernanda Ribeiro Passos Cirino[1], **Juliana Silvestre**[1], **Maria Luísa Raso**[1], **Thayris Gabriela**[1]

[1]Instituto de Ciências Exatas e Informática
Pontifícia Universidade Católica de Minas Gerais
Belo Horizonte – Minas Gerais – Brazil

fcirino@sga.pucminas.br, juliana.silvestresilva@hotmail.com

maria.raso@sga.pucminas.br, tgfrodrigues@sga.pucminas.br

1. Introdução

Na indústria em qualquer lugar do mundo, determinar a quantidade de produtos a serem fabricados é crucial para o processo de produção, porque é essa quantidade que vai ditar se a demanda de mercado será atendida no menor custo possível e sem desperdícios. Nesse contexto, algoritmos de otimização podem auxiliar na determinação dessas quantidades ideais a serem produzidas, considerando restrições como disponibilidade financeira, capacidade de produção e expectativa de demanda, sendo os algoritmos de otimização são ferramentas poderosas na resolução de problemas complexos como esse, de tomada de decisão.

Nesse cenário, métodos como a Programação Linear (PL), a Programação Linear Inteira (PLI) e a Programação Não Linear (PNL) são amplamente utilizados trabalhando em cima desse objetivo fundamental, porque estes algoritmos permitem encontrar soluções ótimas para problemas que envolvem múltiplas variáveis e restrições, proporcionando uma forma eficiente de maximizar lucros ou minimizar custos. A partir disso, a aplicação desses métodos pode transformar a eficiência operacional no cenário da indústria, garantindo que os recursos sejam utilizados da melhor forma possível.

No ramo de bebidas alcóolicas, esse cenário também se aplica, uma vez que existem diversas variáveis na produção que podem mudar tudo, como a utilização e transporte de matéria prima, a relação custo-lucro e a demanda de mercado que cada uma das diversas cervejas tem, por exemplo. Tudo isso requer uma forma otimizada de produção, considerando as quantidades corretas de ingredientes e garrafas produzidas, de forma a evitar perda de ganhos por parte da empresa.

Dada a relevância dessa aplicação, o problema a ser solucionado neste trabalho, portanto, é a otimização de transportes para produção de cervejas. Este artigo tem como objetivo otimizar a compra/entrega da produção cervejeira, considerando suas devidas restrições e variáveis. Para isso, o problema será modelado considerando uma empresa fictícia, utilizando como solução três diferentes algoritmos: Noroeste, Menor Custo e Vogel.

O trabalho é dividido em outras cinco seções, sendo elas: (i) Revisão de literatura e embasamento teórico, (ii) Metodologia, (iii) Estudo Experimental, (iv) Conclusão, abrangendo todos os tópicos necessários para o entendimento do problema e o cumprimento do objetivo proposto e (v) Disponibilização dos artefatos utilizados.

2. Revisão da literatura e embasamento teórico

2.1. Produção de cerveja

O processo da produção de cerveja é amplamente conhecido, e pode ser resumido seguindo os passos da Figura 1, a seguir.

Figura 1. Processo de fabricação de cerveja.

Em suma, os processos são:

1. Maltagem: A cevada é submersa em água e germina, formando o malte (que irá definir o aroma e sabor da cerveja)
2. Moagem: O malte é torrado e, a depender da temperatura, ele trará um sabor diferente à cerveja. Em seguida, ele é moído e misturado em água quente, formando o mosto
3. Fervura: O mosto é aquecido e o lúpulo é acrescentado à ele. Em seguida, o mosto é separado novamente
4. Fermentação: O mosto é resfriado e a levedura é acrescentada à ele. Em seguida, ocorre sua maturação
5. Filtragem: O resultado do processo anterior é filtrado e pasteurizado, se tornando chope. Outra possibilidade é ele ser engarrafado e pasteurizado com água quente, se tornando a cerveja normal

2.2. Algortimos de Otimização

O primeiro algoritmo a ser utilizado neste trabalho é o Simplex, com seu sistema de equações lineares e sua tabela de solução do sistema. O simplex é um método tradicional de otimização utilizado para resolver problemas de programação linear, desenvolvido por George Dantzig na década de 1940 e amplamente utilizado devido à sua eficiência na resolução de problemas lineares com várias variáveis e restrições.

Em primeiro lugar é preciso modelar o problema e transformar as inequações em equações, a partir da introdução de uma variável chamada de folga.

Exemplo:

$$6X1 + 3X2 + 5X3 \leq 500 \tag{1}$$

$$6X1 + 3X2 + 5X3 + Folga = 500 \tag{2}$$

A partir disso, seu funcionamento é dado por:

1. Formulação do Problema: A função objetivo é maximizada (ou minimizada). As restrições são todas expressas como desigualdades (ou igualdades) lineares. As variáveis são todas não negativas.
2. Forma Padrão: Adiciona-se variáveis de folga para transformar as desigualdades em igualdades. A função objetivo é reescrita de modo que todas as variáveis estejam no lado direito da equação.
3. Tabela Simplex Inicial: Linha Z: Representa a função objetivo. Coluna de Variáveis Básicas: Inicialmente, inclui variáveis de folga. Coluna de Coeficientes das Variáveis Básicas: Mostra os coeficientes das variáveis básicas na função objetivo. Matriz de Coeficientes: Contém os coeficientes das variáveis nas restrições. Coluna de Solução: Contém os valores atuais das variáveis básicas.
4. Iteração Simplex: Teste de Otimalidade: Verifica se a solução atual é ótima. Se todos os coeficientes da linha Z são não negativos (para maximização), a solução é ótima. Escolha da Variável Entrante: Escolhe a variável com o coeficiente mais negativo (para maximização) na linha Z para entrar na base. Escolha da Variável Saínte: Usa a regra da razão mínima para determinar qual variável básica será substituída pela variável entrante. Pivotamento: Realiza operações de linha para atualizar a tabela simplex, fazendo a variável entrante tornar-se básica e ajustando todas as outras variáveis.
5. Repetição: O processo de teste de otimalidade, escolha das variáveis entrante e saínte, e pivotamento continua até que a solução ótima seja encontrada.

Além do Simplex, outro algoritmo é utilizado para chegar até uma possível solução ótima. É o algoritmo branch-and-bound. Ele é uma técnica amplamente utilizada para resolver problemas de otimização, especialmente em casos de programação inteira e problemas de otimização combinatória. Ele funciona dividindo o problema em subproblemas menores, descartando subproblemas que não podem produzir soluções melhores do que a melhor solução já encontrada.

Seus passos são:

1. Inicialização: Defina a solução ótima atual como infinita (para problemas de minimização) ou como menos infinita (para problemas de maximização). Coloque o problema original em uma lista de subproblemas a serem resolvidos (geralmente chamada de lista de ”nós ativos”ou ”fronteira de busca”).
2. Expansão (Branching): Retire um subproblema da lista de nós ativos. Este subproblema é o nó atual. Divida (ou ”ramifique”) o subproblema em subproblemas menores. Esta divisão é feita de acordo com as regras do problema, que geralmente envolvem fixar algumas variáveis a determinados valores.
3. Avaliação (Bounding): Para cada subproblema gerado, calcule um limite inferior (para problemas de minimização) ou um limite superior (para problemas de maximização) da melhor solução que pode ser obtida a partir deste subproblema. Compare este limite com a solução ótima atual: Se o limite do subproblema for pior do que a solução ótima atual, descarte o subproblema (poda). Se o limite for melhor ou igual à solução ótima atual, mantenha o subproblema na lista de nós ativos e, se o subproblema representar uma solução completa e viável, atualize a solução ótima atual.
4. Terminação: O algoritmo termina quando a lista de nós ativos estiver vazia, ou seja, quando todos os subproblemas tenham sido resolvidos ou descartados. A solução ótima atual será a solução ótima do problema original.

Um exemplo de como esse algoritmo fica no formato de árvore com podas, é o seguinte:

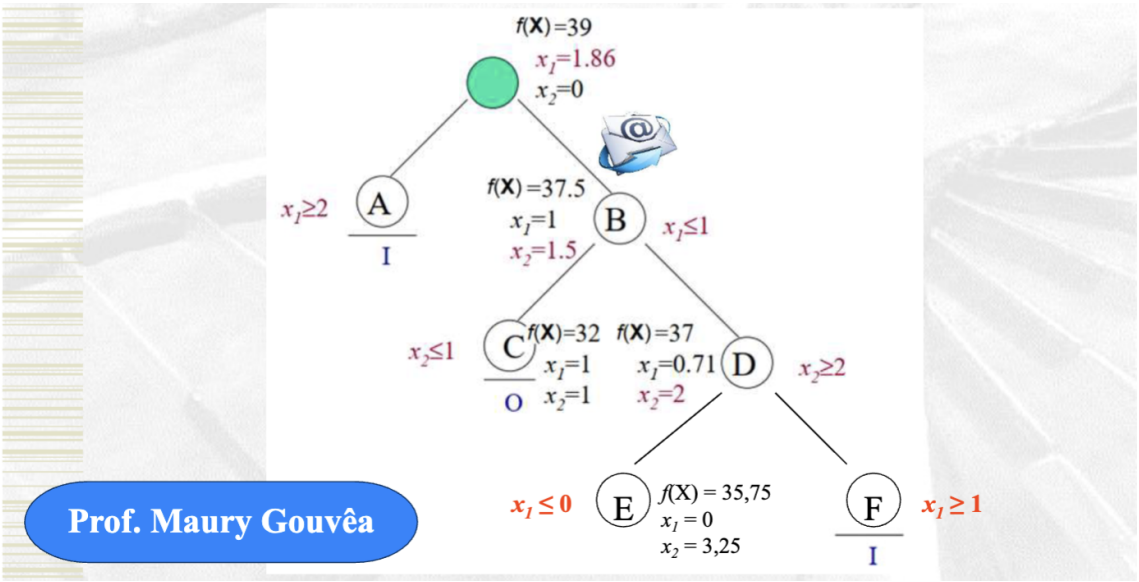

Figura 2. Exemplo de árvore podada do algoritmo branch-and-bound para solução ótima.

2.3. Trabalhos relacionados

Em Costa Pereira (2021), um algoritmo genético foi investigado com o intuito de otimizar a produção de café. O modelo proposto considera variáveis como mão de obra, adubação, colheita e transporte, buscando minimizar os custos totais enquanto maximiza a produtividade. Os resultados demonstram a efetividade do algoritmo genético em reduzir os custos

de produção em comparação com métodos tradicionais, destacando seu potencial como ferramenta para otimizar a gestão da cafeicultura. A partir disso, nota-se que o problema de produção para grãos e bebidas, como é o caso do café e da cerveja, necessitam de muito cuidado ao serem tratados, e a otimização de sistemas pode se aplicar muito bem, dado que neste cenário é comum grandes perdas e prejuízos, que podem ser evitados com contas otimizadas.

Já em Amorim et al. (2015), o método de pesquisa operacional foi utilizado para modelar matematicamente a otimização da produção e maximização dos lucros em uma fábrica de cerveja que produz três tipos diferentes, e apresenta em sua planta industrial múltiplos tanques de fermentação e maturação. São analisados quatro cenários com diferentes valores de capital para compra de matéria prima e as soluções são obtidas utilizando-se a interface open source Gusek. A relaÇão com o trabalho atual se baseia no fato de que, em ambos os artigos, o problema de produção de derveja é tratado com a otimização de sistemas, cada um com um algoitmo diferente, evidenciando a variedade de formas de se solucionar um mesmo problema atavés desta área de estudo.

3. Metodologia

3.1. Descrição do Problema

A empresa Backer descontinuou sua linha de cervejas chamada ”Belorizontina”, após erro em lote de produção que causou intoxicamento em alguns usuários. Dada essa descontinuidade, a capacidade produtiva de suas máquinas entrou em excesso, causando perdas de lucro. Seus donos estão levando em conta a possibilidade de dedicar esse excesso para outras duas cervejas A, B e C já produzidas anteriormente pela empresa.

A capacidade disponível nas máquinas está descrita na tabela a seguir:

Máquina	**Disponível em horas-máquina/semana**
Moagem	500
Aquecimento	350
Pasteurização A	150

Dado essas disponibilidades, o número de horas-máquina exigidas para cada garrafa das cervejas A, B e C são:

Máquina	**Cerveja A**	**Cerveja B**	**Cerveja C**
Moagem	6	3	5
Aquecimento	5	4	3
Pasteurização	3	2	2

Os donos da empresa também estudaram o mercado de cada uma das cervejas A, B e C, e entenderam que o potencial de vendas para as cervejas A e B excedem a taxa de produção máxima e que o potencial de vendas da cerveja C é de 20 unidades por semana. O lucro que cada uma delas gera é de R$ 5,00 por garrafa.

Dadas essas informações, o objetivo da otimização a ser feita é determinar quanto de cada cerveja A, B e C deve ser produzida a partir de agora, para maximizar os lucros.

3.1.1. Equações das restrições

As equações a seguir representam as restrições do problema, considerando as disponibilidades das máquinas e os potenciais de venda. A cerveja A será chamada de X1, a cerveja B será chamada de X2 e a cerveja C será chamada de X3.

Restrição é com relação à Moagem:

$$6X1 + 3X2 + 5X3 \leq 500 \tag{3}$$

Restrição com relação ao Aquecimento:

$$5X1 + 4X2 + 3X3 \leq 350 \tag{4}$$

Restrição com relação à Pasteurização:

$$3X1 + 2X2 + 2X3 \leq 150 \tag{5}$$

Restrição com relação ao potencial de vendas da cerveja C:

$$X3 \leq 20 \tag{6}$$

Positividade da quantidade de garrafas produzidas:

$$X1, X2, X3 >= 0 \tag{7}$$

3.2. Equação de lucro

A equação a seguir representa o lucro, a ser maximizado:

$$\text{MAX(Lucro)} = 5X1 + 5X2 + 5X3 \tag{8}$$

3.3. Método para solução do problema

A escolha dos algorítmos a serem utilizados, conforme descrito na Seção 2, de embasamento teórico, foram o Simplex com sua tabela de solução de equações lineares e o Branch-and-Bound com sua árvore de podas.

Dada essa escolha, e o passo-a-passo descrito de cada um deles, começa-se pelo Simplex, e depois se realiza o Branch-and-Bound para fins comparativos.

4. Estudo experimental

4.1. Implementação do Simplex em Python

O algoritmo Simplex foi implementado em python.

Para a rodagem do programa desenvolvido, entre na pasta do projeto em que está o arquivo simplex.py e, através do terminal, utilize o comando:

python3 simplex.py -A ”[[6,3,5], [5,4,3], [3,2,2], [0,0,1]]” -b ”[500, 350, 150, 20]” -c ”[5,5,5]” -p max

Note que os primeiros Arrays contém os coeficientes das restrições do problema, formando uma matriz. Já o segundo grupo de números contém os termos independentes, resultados das restrições do problema. Por fim, o último Array do comando apresenta os coeficientes presentes na função de lucro a ser maximizada.

Esses valores colocados na linha de comando serão utilizados pelo programa para que a tabela do simplex seja montada devidamente, e seus pivôs sejam encontrados e tratados, seguindo a lógica do algoritmo.

Além disso, neste mesmo comando, os valores dos Arrays podem ser trocados por outros, a fim de solucionar outros problemas que excedem a delimitação deste trabalho.

O resultado da implementação é colocado em um arquivo separado, chamado solution.tex, em que a solução já na identação de Latex é explicitada, conforme mostrado na Seção anterior deste trabalho. A solução colocada nesse arquivo se saída inclui explicação do passo a passo do Simplex e suas tabelas intermediárias até a solução ser encontrada.

A solução dada foi:

$$\max 5x_1 + 5x_2 + 5x_3$$

$$\begin{cases} 6x_1 + 3x_2 + 5x_3 \leq 500 \\ 5x_1 + 4x_2 + 3x_3 \leq 350 \\ 3x_1 + 2x_2 + 2x_3 \leq 150 \\ x_3 \leq 20 \end{cases}$$

Solução: Colocando variáveis de folga e transformando em equações:

$$\begin{cases} 6x_1 + 3x_2 + 5x_3 + s_1 = 500 \\ 5x_1 + 4x_2 + 3x_3 + s_2 = 350 \\ 3x_1 + 2x_2 + 2x_3 + s_3 = 150 \\ x_3 + s_4 = 20 \end{cases}$$

Agora, colocando os valores na tabela do simplex:

$$\left[\begin{array}{cccccccc|c} x_1 & x_2 & x_3 & s_1 & s_2 & s_3 & s_4 & b & \\ \hline 6 & 3 & 5 & 1 & 0 & 0 & 0 & 500 & s_1 \\ 5 & 4 & 3 & 0 & 1 & 0 & 0 & 350 & s_2 \\ 3 & 2 & 2 & 0 & 0 & 1 & 0 & 150 & s_3 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 20 & s_4 \\ \hline -5 & -5 & -5 & 0 & 0 & 0 & 0 & 0 & \end{array}\right]$$

Existem elementos negativos na linha inferior, portanto a solução atual não é a ideal. Portanto, gire para melhorar a solução atual. A variável de entrada é x_1 e a variável de saída é s_3. Execute operações elementares de linha até que o elemento pivô seja 1 e todos os outros elementos na coluna de entrada sejam 0.

$$\left[\begin{array}{cccccccc|c} x_1 & x_2 & x_3 & s_1 & s_2 & s_3 & s_4 & b & \\ \hline 0 & -1 & 1 & 1 & 0 & -2 & 0 & 200 & s_1 \\ 0 & 2/3 & -1/3 & 0 & 1 & -5/3 & 0 & 100 & s_2 \\ 1 & 2/3 & 2/3 & 0 & 0 & 1/3 & 0 & 50 & x_1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 20 & s_4 \\ \hline 0 & -5/3 & -5/3 & 0 & 0 & 5/3 & 0 & 250 & \end{array}\right]$$

Existem elementos negativos na linha inferior, portanto a solução atual não é a ideal. Portanto, gire para melhorar a solução atual. A variável de entrada é x_2 e a variável de saída é x_1. Execute operações elementares de linha até que o elemento pivô seja 1 e todos os outros elementos na coluna de entrada sejam 0.

$$\left[\begin{array}{cccccccc|c} x_1 & x_2 & x_3 & s_1 & s_2 & s_3 & s_4 & b & \\ \hline 3/2 & 0 & 2 & 1 & 0 & -3/2 & 0 & 275 & s_1 \\ -1 & 0 & -1 & 0 & 1 & -2 & 0 & 50 & s_2 \\ 3/2 & 1 & 1 & 0 & 0 & 1/2 & 0 & 75 & x_2 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 20 & s_4 \\ \hline 5/2 & 0 & 0 & 0 & 0 & 5/2 & 0 & 375 & \end{array}\right]$$

Não há elementos negativos na linha inferior, portanto sabemos que a solução é ótima. Assim, a solução é:

$$s_1 = 275, s_2 = 50, s_3 = 0, s_4 = 20, x_1 = 0, x_2 = 75, x_3 = 0, z = 375$$

4.2. Implementação Branch and Bound em Python

O algoritmo Branch and Bound é uma técnica fundamental na área de otimização combinatória, utilizada para resolver problemas onde se busca a melhor solução dentre um conjunto finito de possibilidades, respeitando determinadas restrições. Sua aplicação é ampla, abrangendo desde problemas de escalonamento e roteamento até problemas de design e alocação de recursos.

Dessa maneira, a ideia central deste algoritmo é dividir o problema original em subproblemas menores, denominados ramos, e explorar cada ramo de forma sistemática, atualizando e refinando a melhor solução encontrada até o momento. Este método combina a exploração exaustiva das soluções possíveis (branching) com o uso de limites (bounding) para evitar o processamento desnecessário de ramos que não levariam a uma solução ótima.

No exemplo específico implementado, o algoritmo Branch and Bound é aplicado para maximizar o lucro na produção de três tipos de cerveja (A, B, C), sujeito a múltiplas restrições de produção. Cada tipo de cerveja possui um lucro associado e as restrições incluem limitações na moagem, aquecimento, pasteurização e demanda final de um dos tipos de cerveja. O algoritmo começa com uma solução inicial onde todas as quantidades de produção são zero e explora recursivamente todas as combinações viáveis de quantidades de produção, verificando a viabilidade de cada solução parcial usando a função *is_feasible*.

Durante a execução, o algoritmo atualiza continuamente a melhor solução encontrada *(best_solution)* e seu respectivo lucro *(best_profit)*. Utilizando uma abordagem de profundidade primeiro (DFS), ele ramifica em cada variável de decisão (quantidade de produção de cada tipo de cerveja) até que todas as variáveis tenham sido avaliadas, momento em que avalia se a solução atende a todas as restrições. Se uma solução parcial não for viável, os ramos correspondentes são podados *(cutoff)*, reduzindo o espaço de busca.

Ao final da execução, o algoritmo retorna a melhor combinação de quantidades de produção de cervejas que maximiza o lucro total, respeitando todas as restrições impostas. Esta implementação ilustra a eficácia do método Branch and Bound na resolução de problemas complexos de otimização, oferecendo uma abordagem sistemática e eficiente para encontrar soluções ótimas em um espaço de busca exponencial.

Sua saída foi dada por:

Cerveja A produzida: 0. Cerveja B produzida: 55. Cerveja C produzida: 20. Lucro total: 375

5. Conclusão

O trabalho seguiu o objetivo proposto, utilizando o método Simplex e o Branch-and-Bound de otimização de sistemas para encontrar a solução inteira ótima do problema de utilização de recursos da fábrica de cerveja. Além do método Simplex utilizado, o algoritmo Branch and Bound se destaca por sua capacidade de lidar eficazmente com problemas onde as variáveis de decisão devem ser inteiras, como no caso da produção discreta de diferentes tipos de cerveja em lotes. Ao explorar sistematicamente todas as combinações viáveis de soluções e utilizando cortes para evitar a exploração desnecessária de soluções não promissoras, o Branch and Bound oferece uma abordagem robusta para encontrar soluções ótimas em problemas de otimização inteira. No entanto, existem diversos outros algoritmos que também podem encontrar a solução inteira ótima do problema, trabalhando com os valores de formas diferentes, utilizando mais ou menos recursos e podendo ser mais ou menos simples que a solução implementada neste trabalho.

No exemplo específico analisado, os resultados obtidos pelo Simplex e pelo Branch and Bound divergiram em partes. Enquanto o método Simplex encontrou uma solução onde a produção de cervejas B foi maior (75 unidades de B), resultando em um lucro total de 375, o Branch and Bound alcançou uma solução com menor produção de cerveja B (55 unidades) e maior produção de cerveja C (20 unidades), também com um lucro total de 375 unidades monetárias. Essas diferenças podem ser atribuídas às diferentes abordagens dos algoritmos na exploração do espaço de soluções: o Simplex otimiza de maneira contínua, movendo-se em direção à solução ótima dentro de um espaço contínuo, enquanto o Branch and Bound explora todas as possíveis combinações discretas de soluções, o que pode levar a soluções ótimas diferentes dependendo das características específicas do problema e das suas restrições.Dessa maneira, é crucial entender que ambas as técnicas têm suas vantagens e aplicabilidades específicas. A escolha entre o Simplex e o Branch and Bound depende da estrutura do problema em questão, das características das variáveis de decisão e das restrições envolvidas.

A área de otimização de sistemas é ampla e, portanto, muito rica. As possibilidades são muitas e, nem sempre, a soluÇão ótima será viável na prática. O importante é que

as otimizações sejam buscadas ao máximo, considerando as limitações da realidade e das restrições do problema, que nem sempre são fáceis e, muitas vezes, impedem mesmo a solução ótima de acontecer.

6. Dispobilização dos artefatos

Os códigos do Simplex e do Branch And Bound, juntamente com a versão atualizada desse artigo podem ser encontrados em: `https://github.com/marialuisaraso/TRABALHO-OTIMIZACAO`

Referências

C. P. V. Amorim, L. E. C. Monteiro, P. C. Ribas, et al. Otimização de recursos para maximizar os lucros em uma fábrica de cerveja: utilização da pesquisa operacional para o encontro de valores ótimos de produção. *XVIII Simpósio de Pesquisa Operacional & Logística da Marinha (XVIII SPOLM)*, 2015.

D. Costa Pereira. OtimizaÇÃo do custo de produÇÃo do cafÉ utilizando algoritmo genÉtico. *RECIMA21 - Revista Científica Multidisciplinar - ISSN 2675-6218*, 2(1): 252–266, fev. 2021. doi: 10.47820/recima21.v2i1.73. URL `https://recima21.com.br/index.php/recima21/article/view/73`.