

1

Geração de Colunas e Grasp Reativo para o problema de corte unidimensional

Caio Silas, Júlia Gonzaga

Abstract

This report addresses the One-Dimensional Cutting Problem (PCU), which aims to optimize material usage in cutting processes by minimizing waste. The problem was modeled as an Integer Linear Programming problem, seeking to minimize the number of rolls used. The model was implemented in OPL and tested on two computers. For small datasets, the algorithm was efficient, completing in less than 20 seconds. However, for larger cases, the execution time exceeded 1 hour due to high CPU and memory usage, highlighting hardware limitations in handling more complex problems.

Resumo

Este relatório aborda o Problema de Corte Unidimensional (PCU), que visa otimizar o uso de materiais em processos de corte, minimizando o desperdício. Temos o estudo do problema e sua implementação relacionada ao artigo de pesquisa original. O problema foi modelado como um problema de Programação Linear Inteira, buscando minimizar o número de rolos utilizados. O modelo foi implementado em OPL e testado em dois computadores. Para pequenos conjuntos de dados, o algoritmo foi eficiente, com execução em menos de 20 segundos. No entanto, em casos maiores o tempo de execução ultrapassou 1 hora devido ao alto consumo de CPU e memória, evidenciando as limitações de hardware para lidar com problemas mais complexos.

1. Introdução

Com a ascensão da globalização, as indústrias enfrentam uma crescente pressão para não apenas aumentar a produção em larga escala, mas também para otimizar seus processos. Esta necessidade de eficiência se traduz em economias significativas de tempo, recursos financeiros e matérias-primas. Dentro desse contexto de busca pela otimização da produção e pela redução de desperdícios, emerge o problema do Corte Unidimensional, que se configura como um desafio fundamental para diversos setores, visando maximizar a utilização dos materiais disponíveis.

2. Descrição do Problema

O problema de corte unidimensional consiste em determinar a forma mais eficiente de dividir peças maiores em itens menores, minimizando assim as perdas de material. Este desafio envolve encontrar a melhor estratégia para realizar cortes de diferentes tamanhos em um único objeto, de maneira que se maximize a utilização dos recursos disponíveis. A otimização nesse contexto é crucial, pois impactos significativos podem ser observados nos custos e na viabilidade econômica da produção.

O objetivo da implementação em código descrita neste relatório é determinar quantas vezes cada padrão de corte deve ser utilizado para atender a uma demanda específica de itens, enquanto minimiza o número total de rolos (ou objetos maiores) utilizados. Este é um problema clássico de otimização de corte unidimensional, onde o objetivo é cortar grandes peças (rolos) para obter peças menores (itens) de forma a atender uma demanda, minimizando o número total de grandes peças cortadas.

3. Modelo Matemático Implementado

Admitindo que são conhecidos os padrões de corte unidimensionais, que garantam atender uma demanda de itens a partir de objetos em estoque em quantidade suficiente, o PCU pode ser descrito como um problema de Programação Linear Inteira. Na formulação apresentada a seguir, tem-se o objetivo de determinar o número de vezes que cada padrão de corte é utilizado, de forma a atender tal demanda, utilizando o menor número possível de objetos.

3.1 Função objetivo

A função objetivo é uma expressão matemática que representa o critério que você deseja otimizar em um problema de otimização. Pode ser uma maximização ou minimização de um valor. Por exemplo, em problemas de corte unidimensional, a função objetivo pode ser maximizar a utilização do material ou minimizar o desperdício. O objetivo é encontrar os valores das variáveis de decisão que levarão ao melhor resultado possível segundo essa função.

No nosso problema, a função objetivo visa minimizar o total de rolos utilizados, segue abaixo a implementação e sua forma matemática.

```
// Função objetivo: Minimizar o número total de rolos utilizados
minimize sum(j in Patterns) x[j];
```

$$\text{Min} \quad \sum_{j=1}^n x_j$$

3.2 Restrições

As restrições são condições que limitam as soluções possíveis em um problema de otimização. Elas definem os limites ou requisitos que devem ser satisfeitos. As restrições podem ser baseadas em recursos disponíveis, capacidades técnicas, demandas de produção ou qualquer outra condição que o problema impõe. Por exemplo, no problema de corte unidimensional, uma restrição poderia ser que a soma dos tamanhos dos itens cortados não pode ultrapassar o tamanho da peça original. Segue abaixo as restrições encontradas para o nosso problema:

- Restrição de Demanda: O somatório A_{ij} (quantidade de itens i produzidos no padrão de corte j) vezes a X_j (quantidade de vezes que um determinado padrão de corte é utilizado) tem que ser maior ou igual a D_i (quantidade da demanda do item i).
- Restrição de integralidade: X_i tem que ser maior ou igual a zero.

```
// Restrições: Atender à demanda de cada item
subject to {
    forall(i in Items) {
        sum(j in Patterns) cuts[j][i] * x[j] >= demand[i]; // A demanda deve ser atendida
    }
}
```

$$\text{s.a:} \quad \sum_{j=1}^n a_{ij} x_j \geq d_i, \quad i = 1, \dots, m \quad x_j \geq 0 \text{ e Inteiro, } j = 1, \dots, n$$

3.3 Instâncias

Instâncias referem-se a exemplos específicos ou conjuntos de dados dentro do contexto de um problema de otimização. Cada instância pode ter diferentes parâmetros, como tamanhos de peças, pedidos de corte e restrições. Essas instâncias são usadas para testar algoritmos de otimização e verificar seu desempenho sob diferentes condições. Assim, uma instância do problema de corte unidimensional poderia ser um cenário em que se tem

uma peça de 100 unidades de comprimento e se deseja realizar cortes em tamanhos de 10, 20 e 30 unidades. No nosso problema, identificamos como instâncias:

- M conjunto total de itens distintos;
- N conjunto total dos padrões corte.
- A_{ij} Matriz de padrão de corte i pertence a M e j pertence a N ;
- D_i é a quantidade demandada do item i , i pertence a M ;

3.4 Variáveis de decisão

As variáveis de decisão são os elementos que você pode controlar ou escolher em um problema de otimização. Elas representam as quantidades que você irá determinar no processo de otimização para alcançar a melhor solução. No caso do problema de corte unidimensional, as variáveis de decisão poderiam incluir quantos cortes de cada tamanho devem ser feitos a partir da peça original. No nosso problema, identificamos como variáveis de decisão:

- X_i : é a quantidade de vezes que um padrão de corte foi usado

```
// Variáveis de decisão  
dvar int+ x[Patterns]; // Quantidade de vezes que o padrão de corte j é utilizado
```

4. Resultados

O algoritmo foi implementado em OPL utilizando a IDE CPLEX Studio 22.1.1, disponibilizada pelo professor Pablo Luiz Araújo Munhoz no Moodle. Os testes foram realizados em dois computadores diferentes um com processador Ryzen 5 3600 (6 cores, 12 threads, 3,6 GHz), 16 GB de RAM e Windows 10 (64 bits) e outro com Intel(R) Core(TM) i5-10300H CPU (2.50GHz) 8,00 GB de RAM.

Os resultados com os dados de entrada foram satisfatórios, pois a linguagem OPL é adequada para esse tipo de problema. A execução dos dois modelos levou menos de 20 segundos. Porém, com dados maiores, como 1500 padrões de corte e 1000 itens, houve consumo superior a 80% da CPU e o tempo de execução chegou a 1 hora em ambas máquinas.

```
// solution (optimal) with objective 19  
Número total de rolos utilizados: 19  
Padrão 1: utilizado 2 vezes.  
Padrão 4: utilizado 2 vezes.  
Padrão 5: utilizado 10 vezes.  
Padrão 6: utilizado 5 vezes.
```

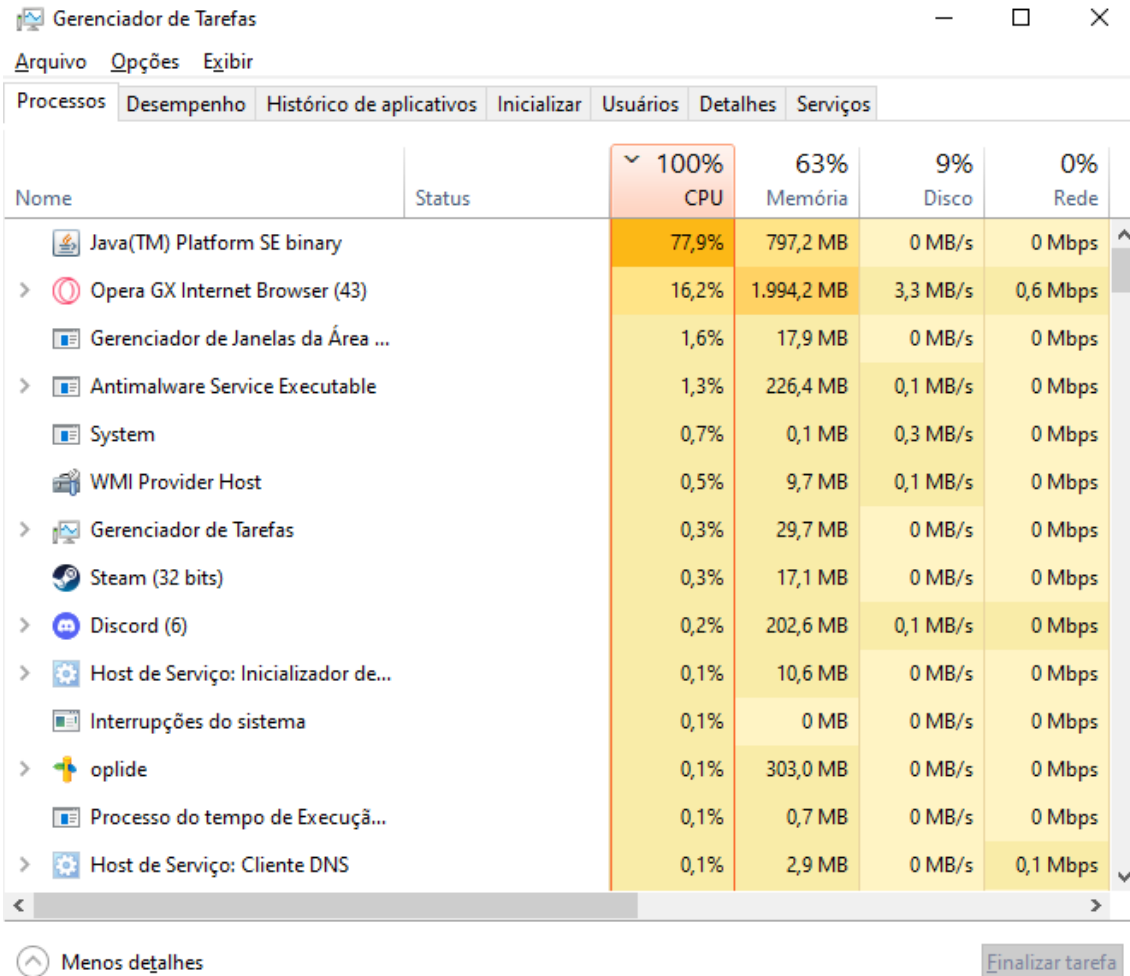


A comparação com o artigo escolhido não foi possível (seria injusta) devido às diferenças na base de dados e no algoritmo utilizado. No entanto, o modelo PCU se mostrou eficiente

para casos simples e médios, com poucas restrições, focando em minimizar a utilização de padrões de corte.

5. Dificuldades e Desafios

Um dos principais desafios encontrados durante o processo de otimização foi o alto consumo de processamento (CPU e memória) pelo código implementado. Conforme evidenciado na imagem fornecida, o programa exigiu uma quantidade significativa dos recursos computacionais disponíveis, o que afetou diretamente o desempenho e a eficiência do processo.



The screenshot shows the Windows Task Manager window with the 'Processos' tab selected. The 'Desempenho' sub-tab is active, displaying a summary of system performance: 100% CPU, 63% Memória, 9% Disco, and 0% Rede. Below this, a list of running processes is shown. The 'Java(TM) Platform SE binary' process is highlighted in orange, indicating it is the most resource-intensive. It is using 77.9% of the CPU and 797.2 MB of memory. Other processes like 'Opera GX Internet Browser' and 'Gerenciador de Janelas da Área ...' are also shown with their respective resource usage.

Nome	Status	100% CPU	63% Memória	9% Disco	0% Rede
Java(TM) Platform SE binary		77,9%	797,2 MB	0 MB/s	0 Mbps
> Opera GX Internet Browser (43)		16,2%	1.994,2 MB	3,3 MB/s	0,6 Mbps
Gerenciador de Janelas da Área ...		1,6%	17,9 MB	0 MB/s	0 Mbps
> Antimalware Service Executable		1,3%	226,4 MB	0,1 MB/s	0 Mbps
System		0,7%	0,1 MB	0,3 MB/s	0 Mbps
WMI Provider Host		0,5%	9,7 MB	0,1 MB/s	0 Mbps
> Gerenciador de Tarefas		0,3%	29,7 MB	0 MB/s	0 Mbps
Steam (32 bits)		0,3%	17,1 MB	0 MB/s	0 Mbps
> Discord (6)		0,2%	202,6 MB	0,1 MB/s	0 Mbps
> Host de Serviço: Inicializador de...		0,1%	10,6 MB	0 MB/s	0 Mbps
Interrupções do sistema		0,1%	0 MB	0 MB/s	0 Mbps
> oplide		0,1%	303,0 MB	0 MB/s	0 Mbps
Processo do tempo de Execuçã...		0,1%	0,7 MB	0 MB/s	0 Mbps
> Host de Serviço: Cliente DNS		0,1%	2,9 MB	0 MB/s	0,1 Mbps

O problema se intensificou ao aumentar o número de padrões de corte e itens a serem processados. Com 1500 padrões de corte e 1000 itens, a execução do código ultrapassou 1 hora, sem que fosse possível visualizar o resultado final. Esse desempenho insatisfatório pode ser atribuído tanto à complexidade algorítmica quanto às limitações da máquina utilizada para os testes.

Iteration	Time	Nodes	Branches	Cuts	Nodes in queue	Nodes in memory
1	0.0000	173,8880	173,8880	173,8880	173,8880	173,8880
2	0.0000	173,8880	173,8880	173,8880	173,8880	173,8880
3	0.0000	173,8880	173,8880	173,8880	173,8880	173,8880
4	0.0000	173,8880	173,8880	173,8880	173,8880	173,8880
5	0.0000	173,8880	173,8880	173,8880	173,8880	173,8880
6	0.0000	173,8880	173,8880	173,8880	173,8880	173,8880
7	0.0000	173,8880	173,8880	173,8880	173,8880	173,8880
8	0.0000	173,8880	173,8880	173,8880	173,8880	173,8880
9	0.0000	173,8880	173,8880	173,8880	173,8880	173,8880
10	0.0000	173,8880	173,8880	173,8880	173,8880	173,8880
11	0.0000	173,8880	173,8880	173,8880	173,8880	173,8880
12	0.0000	173,8880	173,8880	173,8880	173,8880	173,8880

A limitação de recursos, como a quantidade de memória RAM e o processador de médio desempenho, provavelmente contribuiu para o longo tempo de execução do código. A máquina teve dificuldades em lidar com o grande volume de dados e as operações computacionais intensas, o que levou a um consumo exacerbado de memória e processamento.

6. Conclusão

O estudo do Problema de Corte Unidimensional (PCU) baseado no artigo “Geração de Colunas e Grasp Reativo para o problema de corte unidimensional” dos autores João Vitor Bernardo Moreira, e André Soares Velasco destacou a importância da otimização de processos industriais para reduzir o desperdício de materiais e aumentar a eficiência produtiva. A modelagem matemática, implementada em OPL com uso da IDE CPLEX Studio, provou ser eficaz para instâncias de pequeno e médio porte, apresentando resultados satisfatórios em menos de 20 segundos. No entanto, ao lidar com volumes maiores de dados, o modelo enfrentou desafios significativos relacionados ao consumo excessivo de CPU e memória, resultando em longos tempos de execução.

Esses desafios revelam as limitações das máquinas utilizadas para testes, que, devido à restrição de recursos como memória RAM e capacidade de processamento, tiveram dificuldades em processar instâncias maiores. Apesar dessas limitações, o modelo demonstrou seu potencial para casos de menor complexidade, indicando que a otimização, quando devidamente ajustada e aplicada, pode ser uma ferramenta poderosa para a indústria.

Portanto, futuras pesquisas e melhorias devem se concentrar na otimização do código e no uso de equipamentos mais robustos, para lidar melhor com instâncias complexas e de maior escala, maximizando o potencial dos métodos de otimização em cenários industriais.

Referências

da Silva, João Vitor Bernardo Moreira, e André Soares Velasco. "GERAÇÃO COLUNAS E GRASP REATIVO PARA O PROBLEMA DE CORTE UNIDIMENSIONAL." (2021).

NOGUEIRA, Lucas Fernandes . *O problema de corte de estoque unidimensional*. 2018. Dissertação (Bacharelado Ciência da Computação) – Universidade Estadual Paulista, Guaratinguetá, 2018. Disponível em: <https://repositorio.unesp.br/server/api/core/bitstreams/bbc3bc7c-6a92-4bc6-9af5-195f275fa03e/content>.

VASSALO, Lucas Gennari. *Problema de corte de estoque unidimensional*. 2018.
Dissertação – UNICAMP, 2018. Disponível em:
<https://www.ime.unicamp.br/~mac/db/2018-1S-138733.pdf>.