

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**  
**KHOA TOÁN - CƠ - TIN HỌC**



**BÁO CÁO SẢN PHẨM**  
**LẬP TRÌNH NÂNG CAO**

*Giảng viên: LÊ TRỌNG VĨNH*

**Đề tài:**

**CUTTING STOCK PROBLEM**

**Thành viên**

Lê Thị Thùy Dung – 20001895

Trần Ngọc Hải – 20001911

Lưu Hiếu Huy – 20001926

**HÀ NỘI - 2022**

# LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi lời cảm ơn sâu sắc đến giảng viên bộ môn - Thầy Lê Trọng Vĩnh. Thầy đã giảng dạy, truyền đạt, chia sẻ cho chúng em những kiến thức quý báu trong suốt học kì qua. Nhờ sự giúp đỡ của thầy mà chúng em được tiếp thu những kiến thức rất hữu ích cả trong và ngoài chuyên ngành.

Vì thời gian làm bài có hạn nên nhóm không tránh khỏi những thiếu sót. Chúng em rất mong nhận được lời nhận xét từ thầy để cải thiện hơn đồng thời được bổ sung, nâng cao kiến thức của bản thân.

Chúng em xin chân thành cảm ơn thầy vì những kiến thức bổ ích trong thời gian học tập vừa qua. Kính chúc thầy thật nhiều sức khỏe, luôn vui vẻ, hạnh phúc, thành công trong cuộc sống.

# Mục lục

I.	Tổng Quan . . . . .	3
1.	Giới thiệu bài toán . . . . .	3
2.	Mô tả bài toán . . . . .	3
3.	Giải bài toán bằng phương pháp Column Generation	3
3.1	Column Generation là gì . . . . .	3
3.2	Giải Cutting Stock bằng Column Generation	3
4.	Ví dụ cụ thể . . . . .	4
II.	Xây dựng trang web giải bài toán Cutting Stock một chiều .	5
1.	Công nghệ sử dụng . . . . .	5
1.1	Back End: Python . . . . .	5
1.2	Front End: Django HTML. . . . .	6
2.	Triển khai trang web Cutting Stock Problem. . . . .	6
2.1	Cách thức hoạt động của trang web. . . . .	6
2.2	Tiếp nhận, xử lý và trả về dữ liệu cho người dùng. . . . .	7
3.	Sử dụng trang web Cutting Stock Problem . . . . .	11

# I. Tổng Quan

## 1. Giới thiệu bài toán

Cutting Stock là bài toán tối ưu kinh điển thuộc lớp bài toán NP-Hard. Bài toán Cutting Stock có rất nhiều ứng dụng trong thực tế, điển hình là trong hệ thống phục vụ sản xuất tại các nhà máy/công ty sản xuất vật liệu xây dựng. Bài toán đã và đang được nghiên cứu bởi nhiều nhà khoa học trong và ngoài nước. Bài toán Cutting Stock được nảy sinh từ nhiều ứng dụng trong các nhà máy công nghiệp.

Ví dụ: Một nhà máy giấy sản xuất các cuộn giấy thô có chiều dài cố định có thể cắt thành các cuộn nhỏ hơn theo các yêu cầu về số lượng và chiều dài khác nhau của khách hàng. Xác định phương án cắt để số lượng giấy lãng phí là ít nhất, hoặc đúng hơn, số cuộn giấy cần cắt là nhỏ nhất. Đây là một bài toán tối ưu, hay cụ thể hơn là một bài toán quy hoạch tuyến tính nguyên.

## 2. Mô tả bài toán

Bài toán cắt vật tư một chiều với một loại vật liệu thô (bài toán kinh điển) được xác định bởi các dữ liệu sau:  $(m, L, l = (l_1, \dots, l_m), b = (b_1, \dots, b_m))$ , trong đó  $L$  là bề rộng của tấm vật liệu thô,  $m$  là số dạng vật liệu thành phẩm được cắt từ vật liệu thô và đối với mỗi dạng vật liệu thành phẩm  $j$ ,  $l_j$  là bề rộng và  $b_j$  là đơn hàng cho loại vật liệu thành phẩm đó. Bài toán đặt ra là tìm cách cắt sao cho số lượng tấm vật liệu thô sử dụng là ít nhất mà vẫn đáp ứng được đơn hàng.

## 3. Giải bài toán bằng phương pháp Column Generation

### 3.1 Column Generation là gì

Column Generation là một kỹ thuật để giải các bài toán lập trình số nguyên (hỗn hợp) với số lượng biến hoặc cột lớn. Kỹ thuật này lần đầu tiên được áp dụng cho vấn đề lớn trong thực tế đó là Cutting Stock của Gilmore và Gomory. Kể từ đó một số nhà nghiên cứu đã áp dụng kỹ thuật tạo cột cho nhiều ứng dụng thực tế.

### 3.2 Giải Cutting Stock bằng Column Generation

Giả sử cần cắt các cuộn giấy có chiều dài là  $L$  thành  $x_i$ , ( $i = 1 \dots n$ ) đoạn, mỗi đoạn có chiều dài  $l_i$ .

Các phương án cắt khác nhau đều nhằm xác định được số lượng các đoạn

$n_i$  sao cho  $\sum_{i=1}^n l_i x_i$  lớn nhất, hay  $L - \sum_{i=1}^n l_i x_i$  nhỏ nhất.

Mỗi quan hệ số lượng các thanh được cắt ra từ cuộn giấy thô cho trước là quan hệ tuyến tính, khi đó ta sử dụng bài toán quy hoạch tổng quát như sau:

Tìm max, min của hàm mục tiêu

$$z = \sum_{j=1}^n c_j x_j$$

với các ràng buộc  $\sum_{j=1}^n a_{ij} x_j (\leq, =, \geq), i = 1 \dots m, x_j \geq 0, j = 1 \dots n, c$  là vectơ hệ số hàm mục tiêu.

Từ đây ra xây dựng được ma trận hệ số các điều kiện ràng buộc

$$\begin{matrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{matrix}$$

Sau đây ta giải bài toán quy hoạch nguyên để tìm ra phương án cắt tối ưu.

## 4. Ví dụ cụ thể

Cho một số lượng lớn các thanh gỗ có độ dài  $W$ , xác định một cách cắt các thanh gỗ này để tạo ra  $b_i$  đơn vị có độ dài  $w_i$  với  $i = 1, 2, \dots, m$  và số lượng thanh gỗ cần dùng là nhỏ nhất.

Ví dụ:  $W = 15, m = 3$ . Cắt các thanh gỗ dài 15m để đáp ứng đủ:

160 thanh gỗ dài 4m  
100 thanh gỗ dài 6m  
200 thanh gỗ dài 7m

Bài toán trên có các cách cắt như sau:

4m	6m	7m	lãng phí	
3	0	0	3m	
2	1	0	1m	
1	0	1	4m	lãng phí cao
0	2	0	3m	
0	1	1	2m	
1	1	0	5m	lãng phí cao

Tại bảng này, ta thấy có những cách cắt có độ lãng phí cao, ta phải loại bỏ các cách cắt đấy, chỉ giữ lại các cách cắt có độ lãng phí  $\leq \min w_i$

4m	6m	7m	lãng phí
3	0	0	3m
2	1	0	1m
2	0	1	0m
0	2	0	3m
0	1	1	2m
0	0	2	1m

Ta thêm các biến  $x_i$  vào mỗi cách cắt

	4m	6m	7m	lãng phí
$x_1$	3	0	0	3m
$x_2$	2	1	0	1m
$x_3$	2	0	1	0m
$x_4$	0	2	0	3m
$x_5$	0	1	1	2m
$x_6$	0	0	2	1m

Từ bảng trên, ta có thể mô hình hoá bài toán trên như sau:

$$\begin{aligned}
 &\min x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \\
 &\begin{array}{rcl}
 3x_1 & +2x_2 & +2x_3 \\
 & x_2 & +2x_4 +x_5 \\
 & & x_3 +x_5 +2x_6
 \end{array} \begin{array}{l} \geq 160 \\ \geq 100 \\ \geq 200 \end{array}
 \end{aligned}$$

## II. Xây dựng trang web giải bài toán Cutting Stock một chiều

### 1. Công nghệ sử dụng

#### 1.1 Back End: Python

- Framework Django: Django là một framework được viết bằng Python, được thiết kế để đáp ứng nhu cầu phát triển các website an toàn và dễ bảo trì. Django cho phép developer kiểm soát toàn bộ (hoặc một phần) quá trình phát triển website thông qua một nền tảng duy nhất, từ đó, giúp bạn tiết kiệm thời gian và chi phí.

- Thư viện gurobipy: Gurobi là một thư viện phần mềm tối ưu hóa toán học được phát triển bởi Gurobi Optimization, LLC. Nó được coi là một trong những phần mềm tối ưu hóa đa dạng nhất, có thể giải quyết nhiều loại vấn đề. Chúng bao gồm lập trình tuyến tính, lập trình tuyến tính hỗn hợp nguyên, lập trình bậc hai, lập trình bậc hai hỗn hợp nguyên, lập trình ràng buộc bậc hai và lập trình ràng buộc bậc hai hỗn hợp nguyên. Nó có thể truy cập được trong thư viện Python PuLP thông qua API.

## 1.2 Front End: Django HTML.

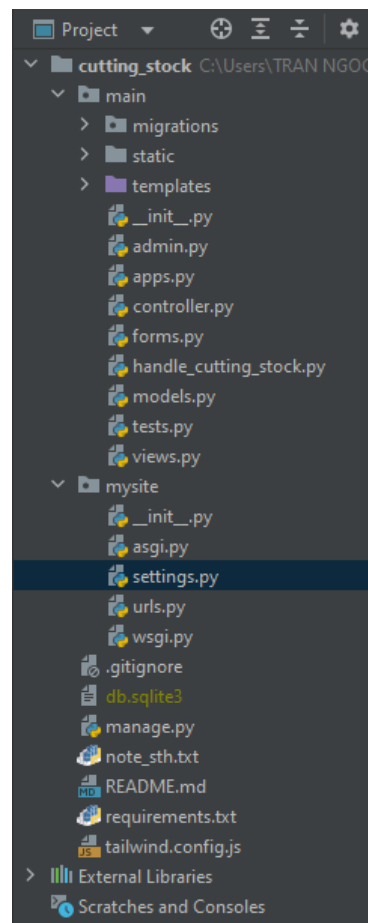
- Tailwind CSS.

## 2. Triển khai trang web Cutting Stock Problem.

Trong phần này, chúng ta sẽ tìm hiểu cách thức hoạt động của trang web Cutting Stock với django và làm thế nào để triển khai thuật toán tối ưu cho dữ liệu đầu vào.

### 2.1 Cách thức hoạt động của trang web.

Trước hết, trong project web cutting-stock chúng ta cần quan tâm đến một vài yếu tố chính giúp trang web có thể được triển khai trên server.



- Package **main**, đây là module mà chúng ta sẽ viết tất cả code cho trang web cutting-stock, kể cả front-end hay back-end. Chúng ta sẽ nói kỹ về package này sau.
- Package **mysite**, đây là module mặc định đi kèm với project, nó gồm các cấu hình đã được framework thiết lập trước để trang web có thể thực thi trên server.

Trong mysite, có file **wsgi.py** và **asgi.py**, hiểu một cách đơn giản rằng chúng giúp web server có thể giao tiếp được với ứng dụng được viết bằng Python hoặc framework (ở đây là Django) từ đó có thể xử lý được request của người dùng.

Trong file **setting.py**. Do trang web của chúng ta là một module (**main** package) nên chúng ta sẽ khai báo rằng module **main** cũng là một webapp của project hiện tại trong **INSTALLED\_APPS**.

```

1 INSTALLED_APPS = [
2     'django.contrib.admin',
3     'django.contrib.auth',
4     'django.contrib.contenttypes',
5     'django.contrib.sessions',
6     'django.contrib.messages',
7     'django.contrib.staticfiles',
8     'main',
9 ]

```

Với file **urls.py**, đây là nơi khai báo các url của project. Để request từ người dùng được xử lý bởi module **main** của chúng ta, chúng ta cần chuyển nó tới cho controller của module **main**.

```

1 urlpatterns = [
2     path('admin/', admin.site.urls),
3     path('', include("main.controller")),
4 ]

```

- Python file **manage.py**.

Khi thực thi file này, tức là chúng ta đã khởi động server và lúc này, web-server đó có thể giao tiếp được với Python code được viết trong project. Để thực thi chúng ta sử dụng lệnh: **python manage.py runserver**

## 2.2 Tiếp nhận, xử lý và trả về dữ liệu cho người dùng.

Đây là phần quan trọng nhất của project và toàn bộ quá trình sẽ được thực thi trong **main** package.



Trước hết, hãy nhìn vào controller, nó thể hiện các đường dẫn trong trang web.

```
1 urlpatterns = [  
2     path("", views.home, name="home"),  
3     path("upload/", views.file_upload_optimize, name="file_upload"),  
4     path("filling_form/", views.click_optimize, name="form_operation"),  
5 ]
```

Ta thấy rằng, khi khởi động server với đường dẫn mặc định, server thực hiện gọi đến hàm `file_upload_optimize()` trong views để xử lý.

```
1 def home(request):  
2     form = FileUploadForm()  
3     return render(request, "main/home.html", {'form': form})
```

Trong views, hàm này trả về trang html home. Và trong home, khi người dùng nhập dữ liệu hay cung cấp input, nó sẽ được gọi vào 2 method tương ứng bên dưới (xem thêm trong phần comment code và file `note_sth.txt`) và từ đó chúng ta có được input đầu vào.

- Input đầu vào với cách nhập thủ công (dạng json):  
{ 'avl\_1': ['10'], 'req[0].length': ['6'], 'req[0].count': ['4'], 'req[1].length': ['5'], 'req[1].count': ['2'], 'quantity\_\_row': ['1'] }
- Input đầu vào khi upload file (theo format từ trước):

```
10  
6  4  
5  2
```

Do định dạng dữ liệu đầu vào giữa 2 cách không giống nhau như vậy, nên chúng ta sẽ cần 2 hàm khác nhau là `get_required_panels_by_filling()`, `get_required_panels_by_uploading_file()` để bóc tách dữ liệu mà chúng ta mong muốn.

```
1 def get_required_panels_by_filling(properties):...  
2  
3 def get_cutting_instruction_by_filling(properties):...  
4  
5 def get_required_panels_by_uploading_file():...  
6
```

```

7 def get_cutting_instruction_by_uploading_file():
8     return main(get_required_panels_by_uploading_file())

```

Cả hai hàm này sẽ đều trả về fixed\_length (chiều dài sẵn có), types (mảng các thanh cần cắt) và numberOfTypes (mảng lưu số lượng tương ứng cho các types). Sau khi đã có được dữ liệu cần thiết, ta thực hiện giải bài toán tối ưu hóa.

Đầu tiên, để giải bài toán, ta phải mô hình hoá được bài toán trên. Đầu tiên, ta phải sinh ra được tất cả các pattern(cách cắt) dựa vào yêu cầu của bài toán.

```

1 #hàm solve để sinh ra tất cả các cách cắt(pattern)
2 #types: các yêu cầu cần cắt, được sắp xếp chiều giảm dần của độ lớn
3 #Đầu tiên ta chia độ dài cố định(fixed_length) cho types[0] được c,
4 #ta thực hiện vòng lặp từ i = 0 đến i = c + 1 và và
5 #để quy hàm solve để có thể đưa ra tất cả các pattern
6 def solve(pattern: list, s: list, fixed_length: int, types: list, j: int,
7                                     lowerBound: float):
8     if j == len(types) - 1:
9         s_c = s.copy()
10        while types[j] <= fixed_length:
11            s_c.append(types[j])
12            fixed_length -= types[j]
13            if fixed_length < lowerBound:
14                pattern.append(s_c)
15        return
16    else:
17        c: int = int(fixed_length / types[j])
18        for i in range(0, c + 1):
19            for k in range(0, i):
20                s.append(types[j])
21                solve(pattern, s, fixed_length - types[j] * i, types,
22                    j + 1, lowerBound)
23            for k in range(0, i):
24                s.pop()

```

Sau đây ta có thể mô hình hoá bài toán bằng cách tìm các ràng buộc - A, hàm mục tiêu - obj, các biến - var để giải quyết bài toán

```

1 def get_frequency_of_types_in_patterns(types, pattern, lengthlist):
2     var = []
3
4     for i in range(0, len(pattern)):
5         var.append("x" + str(i + 1)) #Tạo các biến để đưa vào mô hình tối ưu hoá
6
7     a = [[0 for i in range(0, len(pattern))] for j in range(0, int(lengthlist))]
8     print(types)
9     for i in range(0, len(pattern)):
10        for j in range(0, len(pattern[i])):

```

```

11         a[types.index(pattern[i][j])][i] += 1
12
13 # Thể hiện số lần các thanh có chiều dài yêu cầu được cắt trong các
14 # patterns có thể được cắt
15 # đây sẽ được dùng làm các ràng buộc cho mô hình tối ưu (3)
16     print("a[] = ", a)
17     A = []
18 # Lưu các biến tương ứng với số lần của nó để có thể thêm vào
19 # ràng buộc cho mô hình tối ưu (3)
20     for i in range(0, len(a)):
21         A.append({})
22         for j in range(0, len(var)):
23             A[i][var[j]] = a[i][j]
24
25     obj = {i: 1 for i in var} # hàm mục tiêu
26     return A, obj, var

```

Sau đây ta sử dụng thư viện gurobipy để giải bài toán quy hoạch nguyên

```

1 def gurobi_solve(obj, A, x, numberOfType): # min obj sub Ax >= b
2     print("Gurobi Solve")
3     model = Model()
4     # Thêm biến và các điều kiện cho biến (nguyên)
5     x_var = model.addVars(x, name="variable", vtype=GRB.INTEGER)
6     # Hàm mục tiêu
7     model.setObjective(quicksum(obj[i] * x_var[i] for i in x), GRB.MINIMIZE)
8     # Các ràng buộc của bài toán
9     for i in range(0, len(A)):
10         model.addConstr(quicksum(A[i][j]*x_var[j] for j in x)>=numberOfType[i])
11     # Giải
12     model.optimize()
13     return model

```

Sau khi có được kết quả trả về, ta đưa nó về dạng json (dict) rồi đẩy lên font-end, hiển thị cho người dùng.  
(views.py)

```

1 fixed_length, (cutting_instruction_list,
2     optimal_value) = get_cutting_instruction_by_filling(properties)
3 list_color = ['red', 'blue', 'green', 'yellow', 'pink',
4     'purple', 'orange', 'brown', 'black']
5
6 # Define context for rendering
7 response_context = {'fixed_length': int(fixed_length), 'optimal_value':
8     int(optimal_value), 'cutting_instruction_list': [{ 'number':
9     int(cutting_instruction['number']), 'type': [{ 'value': int(x), 'color':
10     list_color[cutting_instruction['type'].index(x) % len(list_color)]} for x in
11     cutting_instruction['type']], 'residual': int(cutting_instruction['residual'])}]

```

```

12 for cutting_instruction in cutting_instruction_list],
13         }
14 # Path to CWD
15 cwd = os.getcwd()
16 path_template_result = os.path.join(
17     cwd, 'main', 'templates', 'main', 'result.html')
18 # Render the result page
19 return render(request=request, template_name=path_template_result,
20               context=response_context)

```

### 3. Sử dụng trang web Cutting Stock Problem

Đọc và làm theo hướng dẫn ở file README.md để chạy code

Có 2 cách để sử dụng trang web:

- Tải file.txt

Ấn vào format để xem định dạng của file tải vào

Sau khi tải file chọn Optimize để đưa ra kết quả

**Welcome to our home page**

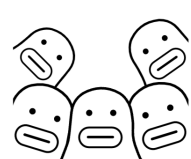
Try our 1d cutting optimizer to find how to cut linear material like lumber, pipes, tubes, bars or beams with minimal material waste. Just specify material length and enter required parts lengths and quantities. Our linear cut calculator will find the best possible solution.

Import your file.txt here

Click on the button to see format input text: Format

Import: Choose File No file chosen

Optimize



- Điền dữ liệu

Điền độ dài của các thanh cần cắt của Length ở Available stock panel

Điền các yêu cầu vào Required panels

Nhập số hàng và ấn Add Row để thêm các cột để nhập yêu cầu

Chọn Optimize để đưa ra kết quả

Available stock panel

Length

Required panels

Length	Quantity	Action
		Delete
		Delete

1

Add Row

Optimize

## Kết quả hiện ra

Quantity là số lượng các thanh gỗ cần cắt

Layout patterns hiển thị ra các cách cắt để số lượng các thanh gỗ là ít nhất. Ở đây hiển thị ra từng cách cắt, và số lượng các thanh cắt theo từng cách.

Sau khi cắt xong, ta tick vào ô bên cạnh để đánh dấu đã cắt xong. Sau khi tích xong hết các thanh, ta sẽ nhận được thông báo đã hoàn thành. Ta có thể ấn New để thực hiện yêu cầu mới.

Cutting Stock

New

Result

Required stocks

Stock length	Quantity
100	38

Layout patterns

Stock length	Repetition	Waste	
100	1x	18	<input type="checkbox"/>
		43	
		39	

Stock length	Repetition	Waste	
100	1x	16	<input type="checkbox"/>
		43	
		41	

# Tài liệu tham khảo

[1] <https://www.youtube.com/watch?v=0918V86Grhc>