ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH


## MÔ HÌNH HÓA TOÁN HỌC (CO2011)

---

### Bài tập lớn

# *CUTTING STOCK PROBLEM*

---

**Giảng viên hướng dẫn**: Ths.Mai Xuân Toàn

**Sinh viên**: Đỗ Huy Minh Dũng *(Lớp DT01)*
Trần Như Mai Anh *(Lớp DT01)*
Phạm Đức Hoàng *(Lớp DT01, **Nhóm trưởng**)*
Lê Sĩ Hoàng *(Lớp DT01)*
Hồ Quốc Đạt *(Lớp DT01)*

Thành phố Hồ Chí Minh, tháng 8 năm 2024

# Mục lục

# Danh sách ký hiệu

$\mathbb{R}$ Tập hợp số thực

$\mathbb{Z}^+$ Tập hợp số nguyên dương

# Danh sách các từ viết tắt

**CSP** Cutting Stock Problem (Bài toán cắt vật tư)

**LP** Linear Programming (Quy hoạch tuyến tính)

**ILP** Integer Linear Programming (Quy hoạch tuyến tính nguyên)

**B&B** Branch and Bound Method (Phương pháp nhánh và cận)

**NP** Nondeterministic polynomial time (Bất định trong thời gian đa thức)

# Danh sách hình ảnh



# Danh sách bảng

# Danh sách thành viên và khối lượng công việc

| STT | Họ và tên | MSSV | Vấn đề | % hoàn thành |
|---|---|---|---|---|
| 1 | Đỗ Huy Minh Dũng | 2210568 | Xây dựng thuật toán và nghiên cứu hiện thực phương pháp suy nghiệm áp dụng nguyên lý tham lam, viết chương trình tạo ra các testcase tham khảo, vá lỗi và cải thiện tốc độ chương trình. | 100% |
| 2 | Trần Như Mai Anh | 2210140 | Xây dựng cơ sở lý thuyết, thu thập các dữ liệu khác, chỉnh sửa báo cáo, giải bài toán bằng tay để kiểm tra độ tin cậy của các giải thuật | 100% |
| 3 | Phạm Đức Hoàng | 2211111 | Xây dựng và hiện thực phương pháp suy nghiệm với giải thuật FFD, phân tích kết quả giải thuật, so sánh các thuật toán, thu thập các dữ liệu liên quan đến bài toán. | 100% |
| 4 | Lê Sĩ Hoàng | 2211082 | Xây dựng thuật toán và nghiên cứu hiện thực phương pháp Simplex, vá lỗi chương trình và cải thiện hiệu suất chạy chương trình, cải tiến độ chính xác của thuật toán. | 100% |
| 5 | Hồ Quốc Đạt | 2210671 | Giới thiệu vấn đề, xây dựng cơ sở lý thuyết, chạy exel đối chiếu các thuật toán, tìm kiếm dữ liệu thực tế cho bài toán, so sánh hiệu suất các chương trình. | 100% |

Bảng 1: Danh sách thành viên và khối lượng công việc

# 1 Giới thiệu vấn đề

Trong quá trình khai thác và sản xuất nội thất làm từ gỗ, công ty Furthen là doanh nghiệp startup trẻ quyết định sản xuất các sản phẩm giường, bàn được làm từ thân gỗ giáng hương. Cây gỗ giáng hương khi trưởng thành có thể cao đến 20 - 30m, các thành phẩm từ gỗ nguyên khối sẽ có giá trị cao hơn so với các sản phẩm làm từ gỗ ép hay gỗ ghép.

Công ty Furthen đã ký hợp đồng với các doanh nghiệp nhằm cung cấp một số lượng lớn bàn, ghế và giường gỗ nguyên khối làm từ cây giáng hương. Biết rằng chiều dài, và số lượng nguyên liệu được cung cấp theo **bảng 2**, số lượng và kích thước thân gỗ cần cắt được cung cấp theo **bảng 3**.

| **Phân loại** | **Chiều dài (mét)** | **Giá thu mua (Triệu đồng)** |
|---|---|---|
| **Cây dưới 15 năm tuổi (Phân loại M)** | **16** | **9** |
| **Cây từ 15 đến dưới 20 năm tuổi (Phân loại L)** | **20** | **13** |
| **Cây 20 năm tuổi trở lên (Phân loại SL)** | **27** | **20** |

Bảng 2: Bảng phân loại và giá thu mua đối tác cung cấp gỗ

| Chiều dài (mét) | 4 | 5 | 9 | 12 | 13 | 21 |
|---|---|---|---|---|---|---|
| Số lượng | 28 | 300 | 290 | 150 | 187 | 320 |

Bảng 3: Số lượng gỗ cần thiết để sản xuất đơn hàng

Bằng cách nghiên cứu các cách cắt, lựa chọn nguyên liệu thân gỗ phù hợp sao cho số chi tiết cắt ra cần thỏa mãn số lượng lớn hơn hoặc bằng số lượng chi tiết cần để lắp ráp được quy định trong **bảng** 3 mà vẫn đảm bảo được tiền vốn là nhỏ nhất có thể từ đó giúp tối thiểu hóa chi phí mua nguyên vật liệu giúp giảm giá thành sản phẩm là mục đích nghiên cứu của nhóm trong bài tập lớn này.

# 2 Kiến thức liên quan

## 2.1 Quy hoạch tuyến tính (Linear Programming)

### 2.1.1 Định nghĩa

Khái niệm quy hoạch tuyến tính là định nghĩa xuất hiện từ những năm 1950 nhằm đề xuất phương án hay lên thời gian biểu cho các hoạt động như huấn luyện, quản lý chuỗi cung ứng. Khái niệm "Tuyến tính" chỉ ra thông qua các điều kiện ràng buộc tuyến tính, sử dụng các hàm mục tiêu để đánh giá nhằm tìm ra phương án tối ưu cho vấn đề cần giải quyết.

### 2.1.2 Biểu thức

Bài toán quy hoạch tuyến tính tổng quát có dạng như sau:

$$\begin{aligned}
&\min && f(x) = c^T x && && (7)\\
&\text{s.t.} && A_i x = b_i, && i \in I_1 && (1a)\\
& && A_i x \le b_i, && i \in I_2 && (1b)\\
& && A_i x \ge b_i, && i \in I_3 && (1c)\\
& && x_j \le 0, && j \in J_1 && (1d)\\
& && x_j \in R, && j \in J_2 && (1e)\\
& && x_j \ge 0, && j \in J_3 && (1f)
\end{aligned}$$

Với

- $I_1 \subset I; I_2 \subset I; I_3 \subset I; I = \{1, 2, \cdots, m\}; I_1 \cup I_2 \cup I_3 = I; I_i \cap I_k = \emptyset, i \neq k, i, k = 1, 2, 3$
- $J_1 \subset J; J_2 \subset J; J_3 \subset J; J = \{1, 2, \cdots, n\}; J_1 \cup J_2 \cup J_3 = J; J_i \cap J_k = \emptyset, i \neq k, i, k = 1, 2, 3$
- Với **c** là vector cột chứa các hệ số $\begin{bmatrix} c_{11} \\ c_{21} \\ \cdots \\ c_{n1} \end{bmatrix}$.
- Ma trận $\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2n} \\ \cdots & & \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$ là ma trận m x n chứa các phần tử $\mathbf{a_{ij}}$ (i = 1,...,m ; j = 1,...,n )
- **x** là vecto cột với kích thước n x 1 (vector biến)
- **b** là vecto cột hệ số kích thước m x 1 thỏa mãn các ràng buộc (constraints).

Trong bài toán trên:

- f được gọi là hàm mục tiêu
- Mỗi hệ thức ở (1a), (1b), (1c), (1d), (1e), (1f) gọi là một ràng buộc

- Mỗi ràng buộc (1a), (1b), (1c) gọi là ràng buộc cưỡng bức (hay cơ bản).
- Ràng buộc (1d), (1e), (1f) gọi là ràng buộc tự do (hay ràng buộc dấu).

Mỗi vectơ $x = (x_1, x_2, \cdots, x_n) \in R^n$ thỏa mãn mọi ràng buộc của bài toán gọi là một phương án. Tập hợp tất cả các phương án (ký hiệu D) gọi là miền ràng buộc hay miền chấp nhận được. Phương án làm cho hàm mục tiêu đạt cực tiểu hoặc cực đại gọi là phương án tối ưu hay một lời giải bài toán đã cho.

Giải bài toán quy hoạch tuyến tính là tìm phương án tối ưu của bài toán (duy nhất hoặc vô số) hoặc chứng tỏ bài toán vô nghiệm.

**Tổng hợp các dạng của bài toán**

Dạng tổng quát bài toán cực tiểu:

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & Cx \geq d \\ & x \geq 0 \end{aligned} \tag{2}$$

Dạng chính tắc bài toán cực tiểu:

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{3}$$

Dạng tổng quát bài toán cực đại:

$$\begin{aligned} \max_x \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & Cx \leq d \\ & x \geq 0 \end{aligned} \tag{4}$$

Dạng chính tắc bài toán cực đại:

$$\begin{aligned} \max_x \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{5}$$

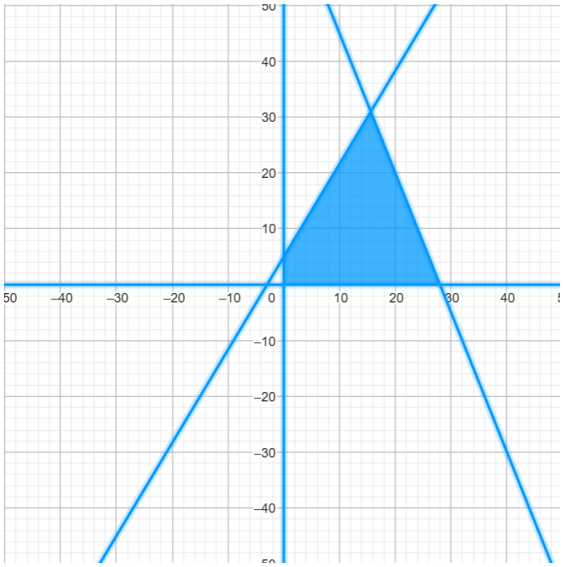
### 2.1.3 Các phương pháp tiếp cận

#### 2.1.3.a Phương pháp hình học

Giả sử ta có bài toán tuyến tính thỏa mô tả sau:

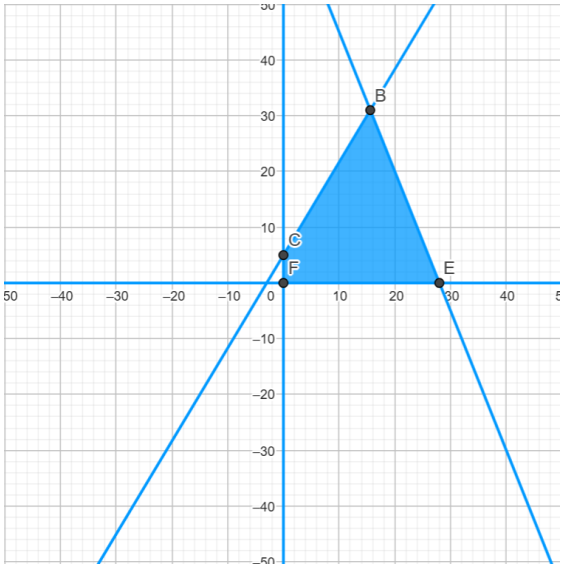
$$\begin{aligned} \max_x \quad & \mathrm{x}_1 + 3\mathrm{x}_2 \\ \text{s.t.} \quad & -\frac{1}{2}x_1 + \frac{1}{5}x_2 \leq 14 \\ & -\frac{1}{3}x_1 + \frac{1}{5}x_2 \leq 1 \\ & \mathrm{x}_1, \mathrm{x}_2 \geq 0 \end{aligned} \tag{6}$$

Phương pháp này thường được sử dụng với các bài toán đơn giản, thường giới hạn trong không gian 2, 3 chiều và số điều kiện ràng buộc nhỏ. Bằng cách vẽ ra các điều kiện ràng buộc, ta sẽ thu được miền không gian khả thi (Feasible region) như sau:




Hình 1: Miền không gian giới hạn bởi các điều kiện ràng buộc

Có thể dễ dàng quan sát, miền khả thi không rỗng và được bao đóng, do đó sẽ tồn tại nghiệm tối ưu (Optimal solution) đối với bài toán cực đại. Vì các điều kiện ràng buộc là các phương trình đường thẳng tuyến tính, do đó cực trị sẽ rơi vào 2 đầu mút của phương trình đường thẳng. Đối với bài toán LP, các điểm cực trị sẽ rơi vào đỉnh của đa giác thuộc miền khả thi.




Hình 2: Các điểm đỉnh đa giác

Với bài toán cực đại như trên, ta có hàm mục tiêu: $Z = x_1 + 3x_2$, ứng với các đỉnh trong miền khả thi ta lần lượt thu được các giá trị:

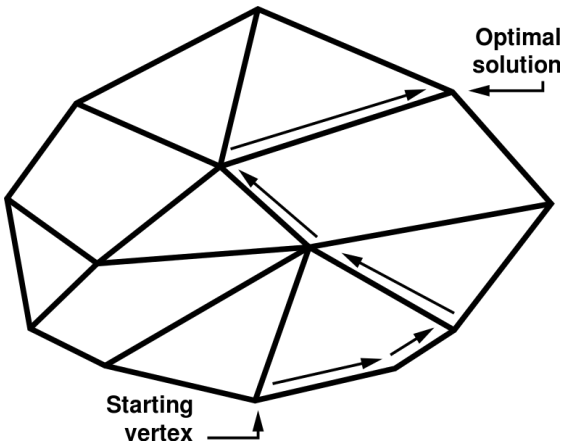
| **Tọa độ đỉnh** | **B(15.6, 31)** | **C(0, 5)** | **E(28, 0)** | **F(0, 0)** |
|---|---|---|---|---|
| $\mathbf{Z = x_1 + 3x_2}$ | 108,6 | 15 | 28 | 0 |

Bảng 4: Bảng giá trị hàm mục tiêu

Để tối đa hóa hàm mục tiêu, ta có thể thấy điểm B chính là nghiệm tối ưu cho bài toán này, ngược lại điểm F lại là nghiệm tối ưu cho bài toán cực tiểu.

#### 2.1.3.b Phương pháp đơn hình (Simplex method)

Để tìm ra nghiệm khả thi tối ưu thỏa mãn, ta cần xác định miền khả thi dựa trên các ràng buộc tuyến tính đã xác định trước. Đối với bài toán cực trị, miền khả thi được bao đóng sẽ luôn tồn tại nghiệm khả thi tối ưu có vị trí nằm trên đỉnh đa giác. Bằng cách thông qua đỉnh bất kỳ là giao điểm của các ràng buộc, phương pháp đơn hình sẽ di chuyển lần lượt đến các điểm tối ưu hơn đến khi tìm ra điểm thỏa mãn. Phương pháp đơn hình được mô hình hóa qua hình minh họa sau:




Hình 3: Mô phỏng trình tự tìm ra nghiệm tối ưu của phương pháp đơn hình

Trình tự giải quyết bài toán tìm nghiệm tối ưu sử dụng phương pháp đơn hình:

- **Bước 1:** Tìm hàm mục tiêu và các điều kiện ràng buộc dạng tổng quát.
- **Bước 2:** Ta tiến hành đưa các điều kiện ràng buộc về dạng chuẩn.
- **Bước 3:** Từ dạng chuẩn, ta lập được ma trận **A** kích thước m x n, vector hệ số **c**. Ta lần lượt chọn các nghiệm cơ sở $\mathbf{x_B}$ (Đỉnh bắt đầu), từ đó lập nên ma trận cơ sở **B**.
- **Bước 4:** Đưa ma trận cơ sở **B** về dạng khả nghịch $\mathbf{B^{-1}}$, do chênh lệch chi phí so với nghiệm tối ưu (reduced cost) có dạng $\mathbf{c_N^T}$ - $\mathbf{c_B^T B^{-1} N}$ vì vậy nếu chênh lệch mang giá trị âm thì $\mathbf{c_B^T x} \leq \mathbf{c_B^T \hat{x}} \Rightarrow$ tồn tại tại nghiệm tối ưu cần tiến hành chọn lại nghiệm cơ sở và thực hiện lại **bước 4** đến khi không tìm được nghiệm tối ưu hoặc tìm thấy nghiệm thỏa mãn.
- **Bước 5:** Giá trị tối ưu khi thực hiện 4 bước sẽ có dạng $\mathbf{c_B^T x}$, kết thúc bài toán.

## 2.2 Quy hoạch tuyến tính nguyên

### 2.2.1 Định nghĩa

Quy hoạch tuyến tính nguyên (Integer linear programming - ILP) là khái niệm chỉ bài toán quy hoạch tuyến tính trong đó tất cả hoặc một phần các biến bị ràng buộc chỉ lấy giá trị nguyên, tương ứng cho từng trường hợp là quy hoạch nguyên hoàn toàn (pure integer programming) và quy hoạch nguyên bộ phận (mixed integer programming - MILP). Lớp bài toán này rất phổ biến trong thực tế, được ứng dụng trong các bài toán sắp xếp lịch trình (Scheduling), sắp xếp vị trí kho (Warehouse location) và bài toán ngân sách (Capital Budgeting).

### 2.2.2 Biểu thức

Bài toán quy hoạch tuyến tính nguyên tổng quát có dạng như sau:

$$\begin{aligned}
&\min && f(x) = c^T x && && (7)\\
&\text{s.t.} && A_i x = b_i, && i \in I_1 && (7a)\\
& && A_i x \le b_i, && i \in I_2 && (7b)\\
& && A_i x \ge b_i, && i \in I_3 && (7c)\\
& && x_j \le 0, && j \in J_1 && (7d)\\
& && x_j \ge 0, && j \in J_2 && (7e)\\
& && x_j \in \mathbb{Z}, && j \in J_3 && (7f)\\
& && x_j \in \mathbb{R}, && j \in J_4 && (7g)
\end{aligned}$$

Với

- $I_1 \subset I; I_2 \subset I; I_3 \subset I; I = \{1, 2, \cdots, m\}; I_1 \cup I_2 \cup I_3 = I; I_i \cap I_k = \emptyset, i \neq k, i, k = 1, 2, 3$
- $J_1 \subset J; J_2 \subset J; J_3 \subset J; J_4 \subset J; J = \{1, 2, \cdots, n\}; J_3 \cup J_4 = J; J_1 \cap J_2 = \emptyset; J_3 \cap J_4 = \emptyset$

Mỗi vectơ $x = (x_1, x_2, \cdots, x_n) \in \mathbb{Z}^n$ thỏa mãn mọi ràng buộc của bài toán gọi là một phương án. Tập hợp tất cả các phương án (ký hiệu D) gọi là miền ràng buộc hay miền chấp nhận được. Phương án làm cho hàm mục tiêu đạt cực tiểu hoặc cực đại gọi là phương án tối ưu hay một lời giải bài toán đã cho.

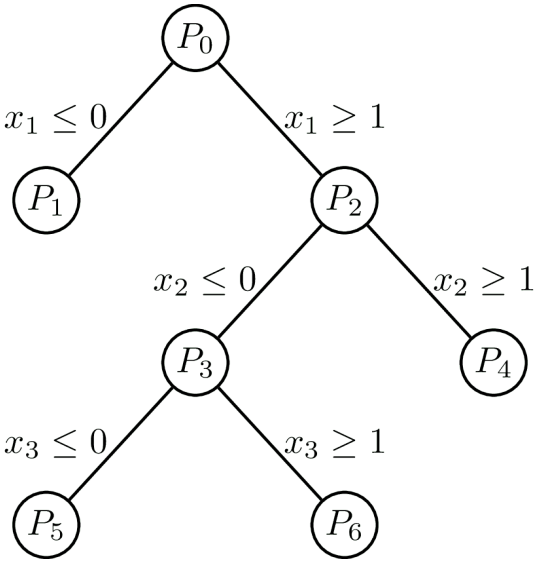
Giải bài toán quy hoạch tuyền tính nguyên là tìm phương án tối ưu của bài toán hoặc chứng tỏ bài toán vô nghiệm.

### 2.2.3 Các phương pháp tiếp cận

#### 2.2.3.a Phương pháp nhánh và cận (Branch and Bound Method)

Phương pháp nhánh và cận (B&B - Branch and Bound) không chỉ giới hạn trong việc giải quyết bài toán quy hoạch tuyến tính nguyên mà còn có thể được áp dụng cho rất nhiều dạng bài toán khác nhau. Đối với các bài toán bất định trong thời gian đa thức (NP - Nondeterministic polynomial time), phường pháp nhánh và cận là phương pháp được sử dụng nhiều nhất để giải các bài toán . Đối với bài toán quy hoạch tuyến tính nguyên, bằng việc chia nhỏ các vật tư thành các đoạn nhỏ hơn để phù hợp với yêu cầu mà vẫn đảm bảo số lượng sử vật phẩm cắt phải là số nguyên, do đó phương án nhánh và cận sẽ được sử dụng để loại bỏ các trường hợp nghiệm không nguyên.

Về nguyên lý hoạt động của phương pháp B&B gần giống như thuật toán quay lui (Backtracking), cây duyệt tìm nghiệm tối ưu của phương pháp sẽ có dạng như sau:




Hình 4: Cây mô hình phương pháp nhánh và cận

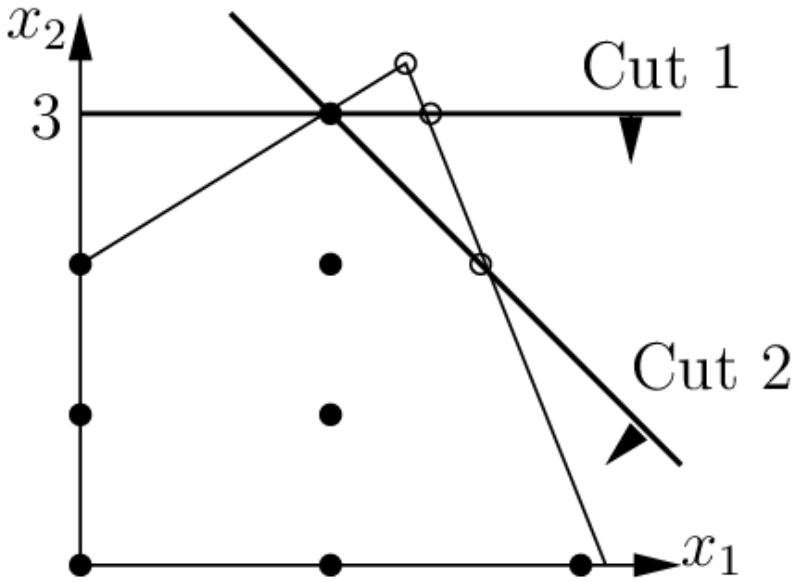
Các nút trong cây đại diện cho các phương án tối ưu, nếu tồn tại biến lựa chọn trong nghiệm tối ưu không nguyên và có phần thập phân gần 0.5 nhất (Most fractional variable) để thực hiện chọn cận trên (UB - Upper Bound) và cận dưới (LB - Lower Bound) cho biến đó. Ta tiến hành lựa chọn điểm đến tiếp theo bằng cách so sánh giá trị hàm mục tiêu của các cận, đối với bài toán cực tiểu thì nút có giá trị nhỏ hơn giá trị tại nút hiện tại sẽ là nút cần nhảy tới (Nghiệm tối ưu hơn) và ngược lại với bài toán cực đại, tiếp tục đến khi tìm được nghiệm thỏa mãn các điều kiện nguyên hoặc không tồn tại nghiệm tối ưu.

#### 2.2.3.b Phương pháp nhát cắt (cutting plane)

Phương pháp nhát cắt (cutting plane) thường được sử dụng để tìm lời giải nguyên cho các bài toán quy hoạch tuyến tính nguyên ILP, quy hoạch tuyến tính nguyên hỗn hợp MILP, cũng như các bài toán tối ưu hóa lồi.

Ý tưởng chính của phương pháp nhát cắt là để giải một bài toán quy hoạch tuyến tính nguyên, chúng ta sẽ giải một chuỗi các bài toán quy hoạch tuyến tính theo các bước như sau:

1. Chuyển bài toán quy hoạch tuyến tính nguyên về bài toán quy hoạch tuyến tính bằng cách bỏ đi các ràng buộc nguyên (gọi là integer-relaxation)
2. Tìm lời giải tối ưu x* cho bài toán quy hoạch tuyến tính: nếu x* là lời giải nguyên (integer solution) thì dừng lại và x* chính là lời giải tối ưu cho bài toán quy hoạch tuyến tính nguyên, ngược lại, tạo ra một nhát cắt bằng cách sử dụng giải thuật (Gomory, 1958), để từ đó tạo ra một ràng buộc mới thoả mãn tất cả các lời giải nguyên khả thi khác x*
3. Thêm ràng buộc mới này vào mô hình, quay trở lại bước 2 và giải lại bài toán




Hình 5: Biểu diễn phương pháp nhát cắt bằng đồ thị

## 2.3 Bài toán cutting stock problem

### 2.3.1 Tổng quan bài toán

Bài toán Cutting stock problem (CSP) là một trong những ứng dụng thực tế đầu tiên của Quy hoạch tuyến tính (LP). Các nghiên cứu về bài toán này được áp dụng rộng rãi không chỉ trong ngành công nghiệp mà còn trong các lĩnh vực khác như máy tính và viễn thông. Bài toán xoay quanh việc tối ưu hóa việc cắt nguyên vật liệu (objects) có kích thước chuẩn, chẳng hạn như cuộn giấy hoặc tấm kim loại, thành các mảnh (items) có kích thước quy định bằng các mẫu cắt phù hợp (pattern). Bài toán CSP là một bài toán tối ưu hóa tổ hợp (combinatorial optimization problem) NP-hard.

Trong bài toán CSP tổng quát, ta có tập nguyên vật liệu $S$ có sẵn và được cắt nhỏ để sản xuất các mảnh thành phẩm. Mỗi nguyên vật liệu $s \in S$ đặc trưng bởi kích thước, hình dạng và chi phí $c_s$ trên mỗi đơn vị và số lượng của từng đơn vị. Mỗi đơn hàng bao gồm tập các mảnh thành phẩm F. Trong đó, thành phẩm $f \in F$ được đặc trưng bởi số lượng yêu cầu (số lượng tối thiểu) $d_f$ và các đặc trưng tương tự nguyên vật liệu được nêu trên. Mẫu cắt là danh sách các mảnh thành phẩm có thể cắt từ một loại nguyên vật liệu cụ thể. Một mẫu cắt đặt trưng bởi nguyên vật liệu $s_p$ được dùng và các số nguyên $a_{pf}$ chỉ số mảnh loại $f$ được cắt từ nguyên vật liệu $s_p$.
Mục tiêu của bài toán, hay tiêu chí cần được tối đa hóa hoặc tối thiểu hóa, thường được chia thành các nhóm sau:

- Tối thiểu hóa số lượng nguyên vật liệu, phần dư, hoặc tổng quát hơn là tối thiểu chi phí sản xuất
- Tối ưu hóa bố cục dựa trên hình dạng mẫu cắt
- Trình tự, sự kết hợp hoặc số lượng mẫu cắt được áp dụng.

Dyckhoff (1990) đã phát triển một hệ thống phân loại cho CSP dựa trên 4 đặc điểm sau:

- Số chiều của bài toán: số chiều tối thiểu để diễn tả hình dạng của mẫu cắt
  - (1) CSP một chiều: áp dụng trong lĩnh vực cắt giấy, đường ống, dây cáp, thanh thép
  - (2) CSP hai chiều: áp dụng trong lĩnh vực sản xuất nội thất, may mặc
  - (3) CSP ba chiều: không có nhiều áp dụng nhưng bài toán liên quan 3D packing problem (bài toán sắp xếp) có ứng dụng rộng rãi trong công nghiệp
- Loại phép gán
  - (B) Các mẫu cắt chứa một số mảnh thành phẩm được chọn sao cho mỗi loại nguyên vật liệu được gán với một mẫu cắt
  - (V) Các mẫu cắt đáp ứng nhu cầu mảnh thành phẩm và được áp dụng trên các loại nguyên vật liệu phù hợp
- Sự phân loại nguyên vật liệu
  - (O) Chỉ có một nguyên vật liệu
  - (I) Các nguyên vật liệu có cùng hình dạng, kích thước
  - (D) Các nguyên vật liệu có hình dạng, kích thước khác nhau
- Sự phân loại mảnh thành phẩm
  - (F) Số lượng thành phẩm ít và có hình dạng, kích thước khác nhau
  - (M) Thành phẩm có số lượng nhiều, và phần lớn trong số chúng có hình dạng, kích thước khác nhau
  - (R) Thành phẩm có số lượng nhiều nhưng ít sự đa dạng về hình dạng, kích thước
  - (C) Các thành phẩm tương đẳng với nhau

### 2.3.2 Lịch sử bài toán

Khái niệm cutting stock problem sớm đã được biết đến là bài toán NP - đầy đủ được giới thiệu vào năm 1939 bởi nhà kinh tế và toán học người Nga Kantorovich. Nguyên nhân ra đời của bài toán được cho rằng trong quá trình sản xuất của đa dạng các ngành hàng, các nhà sản xuất luôn tìm cách để tối ưu cho dây chuyền sản xuất, nhằm cắt giảm lượng nguyên liệu lãng phí hay tăng hiệu suất làm giảm giá tiền. Tiến bộ đầu tiên và quan trọng nhất trong việc giải quyết bài toán CSP là nghiên cứu của Gilmore và Gomory (1961, 1963), trong đó mô tả kỹ thuật tạo mẫu trì hoãn (delayed pattern generation) để giải bài toán CSP một chiều bằng quy hoạch tuyến tính. Kể từ đó, số lượng nghiên cứu trong lĩnh vực ứng dụng này bùng nổ.

Vì đây là bài toán bất định trong thời gian đa thức, độ phức tạp lớn, do đó cho tới nay, đã có rất nhiều phương pháp giải được đề xuất cho bài toán CSP có thể kể đến như phương pháp nhánh và cận hay giải thuật luyện gang (Simulated Annealing). Ngoài ra hướng giải suy nghiệm (Heuristic method) như giải thuật di truyền (Genetic algorithm) hay giải thuật tiến hóa (Evolutionary programming) cũng cho thấy được tính hiệu quả trong việc đưa ra phương án tối ưu.

### 2.3.3 Ứng dụng trong thực tế

CSP được ứng dụng khá rộng rãi trong thực tế, mục tiêu bài toán là tối ưu chi phí, do đó trong hầu hết các ngành sản xuất, chủ đầu tư sẽ cần phải có kiến thức về vấn đề này nhằm tránh lãng phí nguồn vốn. Ngoài ra đối với các lĩnh vực về giao thông, y tế, giáo dục ta sẽ vẫn bắt gặp được sự hiện diện của các vấn đề về chi phí và sắp xếp thời gian mà ta có thông qua quy hoạch tuyến tính hay cụ thể là bài toán cắt vật tư để giải quyết

### 2.3.4 Hướng tiếp cận bài toán

Bài toán CSP thường được tiếp cận bằng một trong hai hoặc kết hợp cả hai cách tiếp cận sau:

- **Quy hoạch tuyến tính**
  Hướng tiếp cận này bắt đầu với một tập hợp các mẫu cắt thỏa yêu cầu và tối ưu hóa qua từng bước cho đến khi tìm được nghiệm tối ưu, thông thường được giải quyết dựa trên relaxation của bài toán CSP. Hai vấn đề chính của hướng tiếp cận này là tạo ra các vectơ cơ sở cho LP; và tìm nghiệm số nguyên từ nghiệm tối ưu của bài toán LP, thường là phân số.

- **Liệt kê**
  Hướng tiếp cận này từ từ thêm vào nghiệm các mẫu cắt phù hợp cho đến khi đạt được một nghiệm tối ưu hoàn chỉnh. Hướng tiếp cận phổ biến thường được triển khai dưới dạng Branch-and-Bound, bằng cách gọi đệ quy chia bài toán lớn thành các bài toán con và tính toán giới hạn trên giá trị hàm mục tiêu. Thuật toán có thể được xem như việc duyệt qua một cây, trong đó đường đi từ gốc của cây đến một trong các lá của nó tương ứng với một giải pháp CSP cụ thể. Vấn đề của cách tiếp cận liệt kê nằm ở số lượng lớn tập con hay bài toán con cần được giải quyết. Tất cả nghiên cứu sử dụng hướng tiếp cận liệt kê đều tập trung vào việc đưa ra các phương pháp tìm kiếm hiệu quả để giảm số lượng bài toán con.

Ngoài ra các kỹ thuật khác nhau có thể kết hợp để giải quyết bài toán CSP một cách hiệu quả.

# 3 Phương pháp chuyển đổi từ bài toán cutting stocks problem một chiều về dạng toán quy hoạch tuyến tính (Linear Programming)

## 3.1 Hàm mục tiêu của bài toán cắt vật tư

Bài toán nhằm tối thiểu tiền vốn mua gỗ được đề cập trong bài nghiên cứu này là bài toán CSP 1 chiều, có phân loại là 1/V/D/R.

Trước hết, cần làm rõ mục tiêu bài toán nhằm tối thiểu tiền vốn mua gỗ khi cắt thành nguyên liệu cung cấp do đó có thể nhận định rằng CSP chính là biến đổi về bài toán quy hoạch tuyến tính cực tiểu. Chi phí thu mua có dạng như sau:

**Chi phí = Tổng số tiền từng loại gỗ × Số lượng gỗ cần mua của từng loại tương ứng**

Do đó cần ta có hàm mục tiêu nhằm tối thiểu chi phí gỗ mua nhằm giảm chi phí sản xuất như sau:

$$\sum_{i=1}^{n} \mathbf{c_i x_i}$$

Với $\mathbf{x_i}$ (i = 1, 2, . . . , n) là số lượng các thanh gỗ được thu mua được quy định trong **bảng** 2 để cắt thành các thanh có chiều dài thỏa mãn **bảng** 3. Do đó đối vấn đề bài toán ta đặt ra sẽ bao gồm 3 biến số lượng lần lượt tương ứng với $\mathbf{x_M}$, $\mathbf{x_L}$ và $\mathbf{x_{SL}}$.

$\mathbf{c} = \begin{bmatrix} 9 \\ 13 \\ 20 \end{bmatrix}$ là vector giá thu mua ứng với từng thân gỗ được phân loại theo **bảng** 2.

Từ đó, dạng của hàm mục tiêu có thể được viết lại như sau:

$$\mathbf{Z = 9x_M + 13x_L + 20x_{SL}}$$

## 3.2 Các điều kiện ràng buộc

Do cần cắt thành các mẫu với số lượng được quy định theo **bảng** 3, do đó số lượng thanh nguyên liệu phân theo kích thước cần đạt tối thiểu thỏa mãn các điều kiện ràng buộc:
Chiều dài ở mỗi mẫu cắt phải luôn đảm bảo không vượt quá chiều dài thanh gỗ nguyên liệu thu mua:

$$\sum_{j=1}^{n} \mathbf{a_{ij} l_j} \leq \mathbf{l_{s_i}}$$

Với $\mathbf{i}$ = 1, 2, 3, . . . , m là đại lượng biểu thị mẫu cắt thứ i.

$\mathbf{l_j}$ là chiều dài thanh gỗ cần cắt tương ứng với chiều dài của $\mathbf{x_j}$ (Chiều dài có thể là 1, 4, 7, 12, 15 hoặc 19).

$\mathbf{l_{s_i}}$ là chiều dài thanh gỗ nhập về được sử dụng để cắt tại mẫu thứ i (Chiều dài tuân theo quy định **bảng** 2).

Số lượng thanh gỗ có chiều dài 1 cần phải đạt tối thiểu 28 thanh, do đó ta có điều kiện ràng buộc:

$$\sum_{i=1}^{m} \mathbf{a_{i1} x_i} \geq 28.$$

Số lượng thanh gỗ có chiều dài 4 cần phải đạt tối thiểu 300 thanh, do đó ta có điều kiện ràng buộc:

$$\sum_{i=1}^{m} \mathbf{a_{i2}x_i} \geq 300.$$

Số lượng thanh gỗ có chiều dài 7 cần phải đạt tối thiểu 290 thanh, do đó ta có điều kiện ràng buộc:

$$\sum_{i=1}^{m} \mathbf{a_{i3}x_i} \geq 290.$$

Số lượng thanh gỗ có chiều dài 12 cần phải đạt tối thiểu 150 thanh, do đó ta có điều kiện ràng buộc:

$$\sum_{i=1}^{m} \mathbf{a_{i4}x_i} \geq 150.$$

Số lượng thanh gỗ có chiều dài 15 cần phải đạt tối thiểu 187 thanh, do đó ta có điều kiện ràng buộc:

$$\sum_{i=1}^{m} \mathbf{a_{i5}x_i} \geq 187.$$

Số lượng thanh gỗ có chiều dài 19 cần phải đạt tối thiểu 320 thanh, do đó ta có điều kiện ràng buộc:

$$\sum_{i=1}^{m} \mathbf{a_{i6}x_i} \geq 320.$$

Tổng hợp điều kiện ta bài toán cắt vật tư có dạng như sau:

$$\begin{aligned} \min_{x} \quad & \textstyle\sum_{i=1}^{n} \mathbf{c_i x_i} \\ \text{s.t.} \quad & \textstyle\sum_{j=1}^{n} \mathbf{a_{ij} x_j} \ \geq b \, (i = 1, 2, 3, ..., m) \\ & \mathbf{x_i} \geq 0 \ (i = 1, 2, ..., n) \end{aligned} \tag{8}$$

Vì số lượng các thân gỗ cần sử phải đảm bảo là số lượng nguyên không âm, do đó ta cần thiết lập điều kiện ràng buộc cho biến số lượng.

$$\mathbf{x_i} \geq 0 \ (\text{i} = 1, 2, ..., \text{n}), \ \mathbf{x_i} \in \mathbb{Z}^+$$

Nhận thấy đây là bài toán quy hoạch tuyến tính nguyên (ILP - Integer Linear Programming) ta cần đưa về bài toán LP sử dụng phương pháp relaxation bỏ đi điều kiện ràng buộc nguyên, như vậy, bằng cách sử dụng các phương pháp như nhánh và cận hay phương pháp cắt ta sẽ tìm được nghiệm tối ưu thỏa mãn giá trị bài toán ILP.

$$\mathbf{x_i} \geq 0 \ (\text{i} = 1, 2, ..., \text{n})$$

Ta có thể đưa điều kiện trên về dạng tổng quát bằng cách bổ sung biến nguyên dương $\mathbf{k_i}$ (Slack variable) tại điều kiện ràng buộc như sau:

$$\begin{aligned} \min_{x} \quad & \textstyle\sum_{i=1}^{n} \mathbf{c_i x_i} \\ \text{s.t.} \quad & \textstyle\sum_{j=1}^{n} \mathbf{a_{ij} x_j} - \mathbf{k_i} \ = b \, (i = 1, 2, 3, ..., m) \\ & \mathbf{x_i}, \mathbf{k_i} \geq 0 \ (i = 1, 2, ..., n) \end{aligned} \tag{9}$$

# 4 Thuật toán

## 4.1 Phương pháp đơn hình (Simplex method)

### 4.1.1 Mô hình hóa

Thuật toán Simplex được sử dụng để giải quyết bài toán Cutting Stock Problem (CSP), một bài toán tối ưu hóa tuyến tính (Linear Programming - LP) nhằm mục tiêu giảm chi phí và lãng phí trong quá trình cắt các thanh nguyên liệu thành các đoạn nhỏ theo yêu cầu. Trong bài toán này, chúng ta cần xác định số lần sử dụng các mẫu cắt (patterns) sao cho tổng chi phí là nhỏ nhất và đồng thời thỏa mãn nhu cầu về chiều dài của từng đoạn.

Dữ liệu đầu vào:

- **stock_lengths**: danh sách chiều dài của các thanh nguyên liệu có sẵn.
- **stock_costs**: chi phí tương ứng với mỗi thanh nguyên liệu.
- **piece_lengths**: danh sách chiều dài các đoạn cần cắt.
- **piece_demands**: nhu cầu về số lượng của mỗi đoạn.

Mục tiêu: Tìm số lần sử dụng mỗi mẫu cắt (pattern) sao cho tổng chi phí là nhỏ nhất và đáp ứng được nhu cầu của từng đoạn cắt.

Biến quyết định:

- **x_i**: số lần sử dụng mẫu cắt thứ i.

Hàm mục tiêu:

- Minimize tổng chi phí sử dụng các mẫu cắt: **Minimize** $\sum_i \textbf{pattern\_costs}[i] \cdot x_i$.

Ràng buộc:

- Đáp ứng nhu cầu về số lượng của mỗi đoạn: **Minimize** $\sum_i \textbf{pattern\_costs}[i][j] \cdot x_i \geq piece\_demands[j], \forall j$.
- Các biến quyết định không âm và là số nguyên: $x_i \geq 0$ **và** $x_i \in Z$.

### 4.1.2 Phân tích thuật toán

Để việc phân tích trở nên dễ dàng hơn, hãy xem xét một ví dụ đơn giản về bài toán One-dimensional Cutting Stock như sau: Giả sử có hai loại thanh sắt A và B với chiều dài lần lượt là 15 và 17 đơn vị. Chúng ta cần cắt các thanh này thành các đoạn có chiều dài 4, 6, và 7 đơn vị. Nhu cầu cho các đoạn này lần lượt là 80, 50, và 100 đơn vị. Hãy tìm cách cắt sao cho chi phí tổng cộng là thấp nhất.

Đầu tiên, chúng ta cần xác định các kiểu cắt (cut patterns) có thể thực hiện từ hai loại thanh sắt ban đầu. Ví dụ, từ thanh A có chiều dài 15 đơn vị, chúng ta có thể cắt được 3 thanh có chiều dài 4 đơn vị và còn thừa 3 đơn vị. Như vậy, chúng ta có thể xây dựng được một kiểu cắt $p_{A1}$ là [3, 0, 0], với $p_{Aij}$ là số lượng thanh thứ $j$ có thể cắt được từ thanh A. Sử dụng phương pháp Backtracking, chúng ta có thể tiếp tục tìm các kiểu cắt còn lại, được biểu diễn trong bảng dưới đây.

| | Pattern | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| $a_4$ | 3 | 0 | 0 | 2 | 2 | 0 | 1 | 1 | 4 |
| $a_6$ | 0 | 2 | 0 | 0 | 1 | 1 | 1 | 2 | 0 |
| $a_7$ | 0 | 0 | 2 | 1 | 0 | 1 | 1 | 0 | 0 |
| Waste | 3 | 3 | 1 | 0 | 1 | 2 | 0 | 1 | 1 |
| Stocks | A | A | A | A | A | A | B | B | B |

Bảng 5: Các kiểu cắt từ thanh sắt A và B

Để các kiểu cắt (pattern) khả thi, ta cần đảm bảo rằng:

$$4a_4 + 6a_6 + 7a_7 \leq 15$$

Với $a_4$, $a_6$, $a_7$ lần lượt là số lượng thanh cắt có chiều dài 4, 6, và 7. Bài toán của chúng ta bây giờ được thu gọn thành việc xác định số lượng thanh mà chúng ta cắt theo các kiểu cắt khác nhau và chúng ta muốn tối thiểu hóa số lượng này. Ta ký hiệu $x_j$ là số lượng thanh được cắt theo kiểu cắt thứ $j$. Mục tiêu của chúng ta là tối thiểu hóa tổng $x_j$, vì vậy chúng ta có:
**min**

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9$$

**s.t**

$$\begin{pmatrix}3\\0\\0\end{pmatrix} x_1 + \begin{pmatrix}0\\2\\0\end{pmatrix} x_2 + \begin{pmatrix}0\\0\\2\end{pmatrix} x_3 + \begin{pmatrix}2\\0\\1\end{pmatrix} x_4 + \begin{pmatrix}2\\1\\0\end{pmatrix} x_5 + \begin{pmatrix}0\\1\\1\end{pmatrix} x_6 + \begin{pmatrix}1\\1\\1\end{pmatrix} x_7 + \begin{pmatrix}1\\2\\0\end{pmatrix} x_8 + \begin{pmatrix}4\\0\\0\end{pmatrix} x_9 \geq \begin{pmatrix}80\\50\\100\end{pmatrix}$$

$$x_1, x_2, \ldots, x_9 \geq 0 \text{ và } x_i \text{ là số nguyên, với } i = 1, 2, \ldots, 9.$$

Để không phải xử lý các biên dư thừa, ta sẽ coi như số lượng thanh sau khi cắt bằng với số lượng thanh chúng ta cần. Điều này có nghĩa là chúng ta đang hướng đến việc đáp ứng nhu cầu một cách chính xác và không mong muốn có bất kỳ phần dư nào. Tuy nhiên khi nhìn vào cách chúng ta xây dựng bài toán, chúng ta có thể gặp rất nhiều mẫu cắt khác nhau. Đặc biệt khi xét đến quy mô thực tiễn, số lượng mẫu có thể quá lớn để chúng ta xử lý trong một bảng đơn. Để tránh trường hợp đó, chúng ta sẽ áp dụng quy trình gọi là sinh cột trì hoãn. Chúng ta bắt đầu với một số lượng mẫu giới hạn và thông thường sẽ là những mẫu đơn giản nhất để bắt đầu. Trong ví dụ này, ta sẽ chọn ba mẫu đầu tiên để bắt đầu. Chúng ta sẽ lập một phiên bản hạn chế của bài toán, chỉ chứa ba cột đầu tiên và tạm thời bỏ qua các cột còn lại. Trong thực tế, chúng ta không biết và cũng không cần biết tổng số mẫu cắt mà chúng ta có thể có. Chúng ta chỉ tập trung vào số lượng mẫu hạn chế và sau đó sẽ chỉ sinh các cột có thể đóng góp vào giải pháp tối ưu. Bằng cách này chúng có thể tiết kiệm đáng kể về yêu cầu không gian và giải quyết bài toán một cách hiểu quả hơn. Dưới đây là phiên bản hạn chế của bài toán chính (Restricted Master Problem - RMP):

$$\text{min: } x_{A1} + x_{A2} + x_{A3}$$

**s.t**

$$\begin{pmatrix}3\\0\\0\end{pmatrix}x_1+\begin{pmatrix}0\\2\\0\end{pmatrix}x_2+\begin{pmatrix}0\\0\\2\end{pmatrix}x_3=\begin{pmatrix}80\\50\\100\end{pmatrix}$$

Như vậy, chúng ta đã có phiên bản hạn chế của bài toán chính, nó chỉ có ba biến và ma trận tương ứng là ma trận chéo nên việc xử lý thực sự rất dễ dàng. Thông thường ta sẽ lưu trữ thông tin về cơ sở tối ưu cho phiên bản hạn chế của bài toán chính dưới dạng sau:

| $C_B^T B^{-1}$ | z | BV |
|---|---|---|
| $B^{-1}$ | $B^{-1}b$ | list of BV |

Bảng 6: Bảng thông tin cơ sở tối ưu

Với bài toán của chúng ta, việc xác định các yếu tố trên rất dễ dàng. Ma trận chéo $B^{-1}$ có giá trị là:

$$B^{-1}=\begin{pmatrix}\frac{1}{3}&0&0\\0&\frac{1}{2}&0\\0&0&\frac{1}{2}\end{pmatrix}$$

Đối với $C_B^T B^{-1}$ ta nhân vector của tất cả giá trị bằng 1 với ma trận $B^{-1}$ sẽ thu được:

$$\begin{pmatrix}\frac{1}{3}&\frac{1}{2}&\frac{1}{2}\end{pmatrix}$$

Sau đó, giá trị của hàm mục tiêu sẽ là:

$$\frac{80}{3}+\frac{50}{2}+\frac{50}{1}=\frac{305}{3}$$

Và thành phần tương ứng với các biên cơ sở $B^{-1}b$ sẽ là:

$$\begin{pmatrix}\frac{80}{3}&25&50\end{pmatrix}^T$$

Đặt tất cả giá trị trên vào bảng 6 ta sẽ có được:

| $\frac{1}{3}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{305}{3}$ | BV |
|---|---|---|---|---|
| $\frac{1}{3}$ | 0 | 0 | $\frac{80}{3}$ | $x_1$ |
| 0 | $\frac{1}{2}$ | 0 | 25 | $x_2$ |
| 0 | 0 | $\frac{1}{2}$ | 50 | $x_3$ |

Bảng 7: Bảng giá trị cơ sở tối ưu

Như vậy ta đã có nghiệm tối ưu của bài toán hạn chế và đó cũng chính là một những nghiệm khả thi trong bài toán chính (RMP). Tuy nhiên, ta vẫn chưa thể khẳng định rằng nghiệm đó

chính là nghiệm tối ưu của bài toán chính hay chưa. Để kiểm tra điều đó, ta cần xem xét coi có nên thêm pattern nào từ LP tổng quát vào cơ sơ hay không. Đầu tiên, ta cần tính chi phí giảm cho một biến không cơ bản $x_j$ bằng công thức:

$$Reduced\_cost = c_N^T - C_B^T B^{-1} N = c_N^T - y^T N = 1 - y^T N_j$$

Trong đó:

- $c_N^T$ là hệ số của biến không cơ bản $x$ trong hàm mục tiêu (ở đây là 1).
- y là nghiệm đối ngẫu tối ưu, được tính từ $C_B^T B^{-1}$.
- $N_j$ là cột tương ứng của ma trận không cơ bản.

Nếu chi phí của một biến $x_j$ là âm thì nó có thể cải thiện nghiệm hiện tại khi được thêm vào cơ sở. Trong trường hợp có nhiều biến $x_j$ thỏa điều kiện, ta sẽ chọn biến có chi phí giảm nhỏ nhất để thêm vào. Khi chi giảm nhỏ nhất đó bằng không thì chúng ta có thể kết luận rằng đã có nghiệm tối ưu cho bài toán gốc. Tất cả quá trình trên được gọi là sinh cột, và để làm nó ta cần tạo ra nghiệm tối ưu cho bài toán này. Tuy nhiên, như đã nó, chúng ta không thực sự muốn tạo ra danh sách tất cả các mẫu cắt một cách tường minh. Thay vào đó, chúng ta có thể coi các phần tử trong vector $A_j$ như là cá biến và cố gắng sinh ra mẫu cắt cần thiết để tối đa hóa biểu thức này. Trong trường hợp này ta có

$$A_j = \begin{pmatrix} a_4 \\ a_6 \\ a_7 \end{pmatrix}$$

và mục tiêu của bài toán con sinh cột trở thành tối đa hóa biểu thức:

$$\text{max: } y_1 \cdot a_4 + y_2 \cdot a_6 + y_3 \cdot a_7$$

$$\text{Với: } 4a_4 + 6a_6 + 7a_7 \leq 15$$

$$a_4, a_6, a_7 \geq 0$$

$$a_4, a_6, a_7 \in Z$$

Khi giải quyết bài toán trên, sẽ có hai khả năng có thể xảy ra. Nếu giá trị mục tiêu tối ưu của bài toán con sinh cột $y^T N_j \leq 1$, khi đó $Reduced\_Cost$ sẽ lớn hơn hoặc bằng 0. Điều này có nghĩa nghiệm tối ưu của bài toán hạn chế (RMP) cũng là nghiệm tối ưu của bài toán. Còn nếu giá trị mục tiêu của bài toán con sinh cột $y^T N_j > 1$, khi đó $Reduced\_Cost$ sẽ âm. Điều này có nghĩa có một biến mới cần được thêm vào cơ sở của bài toán hạn chế. Sau khi thêm một biến mới vào bài toán hạn chế (RMP), ta tiếp tục lặp lại quá trình này cho đến khi tìm được nghiệm tối ưu cho bài toán gốc. Trở lại bài toán, nhìn vào bảng 7 ta xác định được các giá trị $y_1 = \frac{1}{3}$, $y_2 = \frac{1}{2}$, và $y_3 = \frac{1}{2}$. Bài toán sinh cột sẽ trở thành:

$$\text{max: } \frac{1}{3}a_4 + \frac{1}{2}a_6 + \frac{1}{2}a_7$$

$$\text{Với: } 4a_4 + 6a_6 + 7a_7 \leq 15$$

$$a_4, a_6, a_7 \geq 0$$

$$a_4, a_6, a_7 \in Z$$

Vì đã có tất cả các mẫu cắt được liệt kê ở Bảng 5, chúng ta có thể kiểm tra xem mẫu cắt nào có giá trị hàm mục tiêu lớn nhất. Ở đây mẫu cắt số 7, 8 và số 9 đều mang lại giá trị tối ưu cho bài toán. Chúng ta sẽ chọn cột 7 để đưa vào cơ sở với $a_7 = (1, 1, 1)^T$. Giá trị mục tiêu tối ưu cho bài toán sinh cột $w = \frac{1}{3} + \frac{1}{2} + \frac{1}{2} = \frac{4}{3} > 1$. Cột đi vào lúc này sẽ là:

| |
|---|
| $\frac{1}{3}$ |
| $\frac{1}{3}$ |
| $\frac{1}{2}$ |
| $\frac{1}{2}$ |

Bảng 8: Enter Column

Khi áp dụng kiểm tra tỷ lệ (Ratio Test) để xác định hàng nào sẽ rời khỏi cơ sở, ta thực hiện chia các giá trị trong cột tay phải (right-hand side column) cho các giá trị tương ứng trong cột vừa được thêm vào (cột $a_7$).

- Đối với hàng đầu tiên, ta có: $\frac{80}{\frac{1}{3}} = 80$.
- Đối với hàng thứ hai, ta có: $\frac{25}{\frac{1}{2}} = 50$.
- Đối với hàng thứ ba, ta có: $\frac{50}{\frac{1}{2}} = 100$.

Kết quả cho thấy tỷ lệ của hàng thứ hai nhỏ nhất, do đó, hàng thứ hai sẽ rời khỏi cơ sở. Bảng giá trị cơ sở tối ưu lúc này:

| | | | | |
|---|---|---|---|---|
| $\frac{1}{2}$ | $\frac{1}{6}$ | $\frac{1}{2}$ | 85 | BV |
| $\frac{1}{2}$ | $\frac{-1}{3}$ | 0 | 10 | $x_1$ |
| 0 | 1 | 0 | 50 | $x_7$ |
| 0 | $\frac{-1}{2}$ | $\frac{1}{2}$ | 25 | $x_3$ |

Bảng 9: Bảng giá trị cơ sở tối ưu

Chúng ta tiếp tục quay trở lại danh sách các pattern và thực hiện quá trình duyệt cho đến khi kiểm tra hết các pattern hoặc khi giá trị *Reduced_Cost* của những pattern này nhỏ hơn hoặc bằng 1. Khi đó, chúng ta sẽ thu được kết quả tối ưu là:

- Pattern [0,0,2] được dùng 25 lần và được cắt từ thanh có độ dài 15.
- Pattern [4,0,0] được dùng 8 lần và được cắt từ thanh có độ dài 17.
- Pattern [1,1,1] được dùng 50 lần và được cắt từ thanh có độ dài 17.

Trong trường hợp cần tối ưu hóa chi phí, chúng ta chỉ cần nhân giá trị của mỗi thanh với số lượng thanh được cắt theo từng kiểu cắt $j$, sau đó tìm giá trị nhỏ nhất của tổng các giá trị này bằng cách áp dụng các bước đã được trình bày ở trên.

## 4.2 Phương pháp suy nghiệm sử dụng kỹ thuật quay lui (Modified Greedy Heuristic with Backtracking Approach)

### 4.2.1 Mô hình hóa

**Thuật toán này không tiếp cận vấn đề như một bài toán quy hoạch tuyến tính.** Nhóm tác giả phát triển thuật toán bằng cách xem vấn đề là một bài toán lập trình cấu trúc dữ liệu và giải thuật với đầu vào (input) được lưu trữ bằng cấu trúc dữ liệu vector, trong đó:
- Stocks: là vector chứa các loại nguyên liệu có sẵn.
- Stocks_cost: là vector chỉ ra chi phí của mỗi nguyên liệu có sẵn tương ứng (được thêm vào với bài toán tiết kiệm chi phí).
- Orders: là vector chứa các độ dài của các mục cần cắt.
- Demand: là vector chỉ ra số lượng mỗi mục cần được cắt tương ứng.

Đầu ra (output) của bài toán sẽ trả về một biến waste thể hiện cho tổng lượng phế liệu sinh ra từ việc cắt nguyên liệu để đáp ứng nhu cầu của khách hàng. Mục tiêu của bài toán là biến trả về waste này phải mang giá trị nhỏ nhất có thể, nói cách khác thuật toán này sẽ cố gắng sử dụng nguyên liệu một cách hiệu quả nhất để đáp ứng nhu cầu cắt ghép mà không gây lãng phí.

### 4.2.2 Phân tích giải thuật

Để độc giả có thể hiểu rõ hơn về ý tưởng của giải thuật, bài báo cáo sẽ trình bày quá trình hiện thực song song với giải quyết bài toán ví dụ bên dưới:

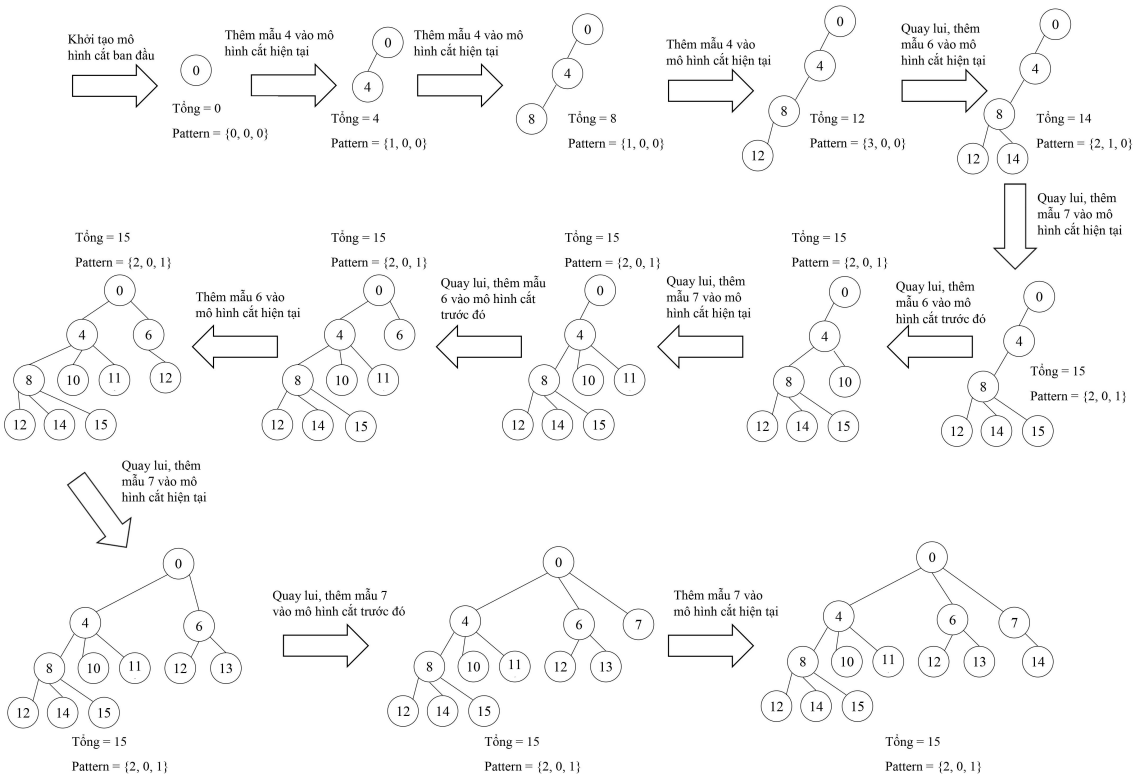
Một công ty sản xuất các thanh kim loại dài 15, 20, 24 đơn vị. Họ nhận được đơn đặt hàng các thanh có chiều dài 4, 6 và 7 đơn vị. Nhu cầu tương ứng cho các thanh này lần lượt là 90, 70, và 110. Mục tiêu là đáp ứng đơn hàng này trong khi giảm thiểu tối đa lượng phế liệu thô.

Đối với một bài toán NP-hard, không thể giải quyết vấn đề chỉ với một hàm (function) duy nhất, vì vậy cần chia nhỏ bài toán này thành nhiều bài toán con và tiến hành giải quyết từng bài toán. Bên cạnh đó, do khối lượng đầu vào của bài toán khá lớn, nhóm sẽ sử dụng lớp (class) để nhận đầu vào và hiện thực hàm cho từng vấn đề nhỏ được chia ra.

Các đầu vào của class bao gồm:
- **Stocks**: vector chứa các loại nguyên liệu có sẵn.
Với ví dụ trên, vector **Stocks** được truyền vào sẽ là {15, 20, 24}
- **Orders**: vector chứa các độ dài các mục cần cắt của đơn hàng.
Với ví dụ trên, vector **Orders** được truyền vào sẽ là {4, 6, 7}
- **Demand**: vector chỉ ra số lượng mỗi mục cần được cắt tương ứng.
Với ví dụ trên, vector **Demand** được truyền vào sẽ là {90, 70, 110}

Để tìm ra cách đáp ứng đơn hàng tiết kiệm nhất, trước tiên chúng ta cần tìm ra cách tận dụng tối đa một thanh nguyên liệu thô. Mục tiêu đầu tiên là tìm ra một tổng gồm các số là phần tử thuộc vector **Orders** sao cho tổng này bằng hoặc nhỏ hơn biến **stock** một giá trị nhỏ nhất (stock là một phần tử của vector **Stocks**). Tổng này sẽ được biểu diễn bằng một vector **Pattern** chỉ ra số lượng mỗi mục cắt đã tạo ra tổng đó.
Xét tới ví dụ, mục tiêu hiện tại là tìm cách cắt các thanh kim loại thô 15, 20, 24 đơn vị thành các mẫu 4, 6, 7 đơn vị sao cho lãng phí là không có hoặc ít nhất.

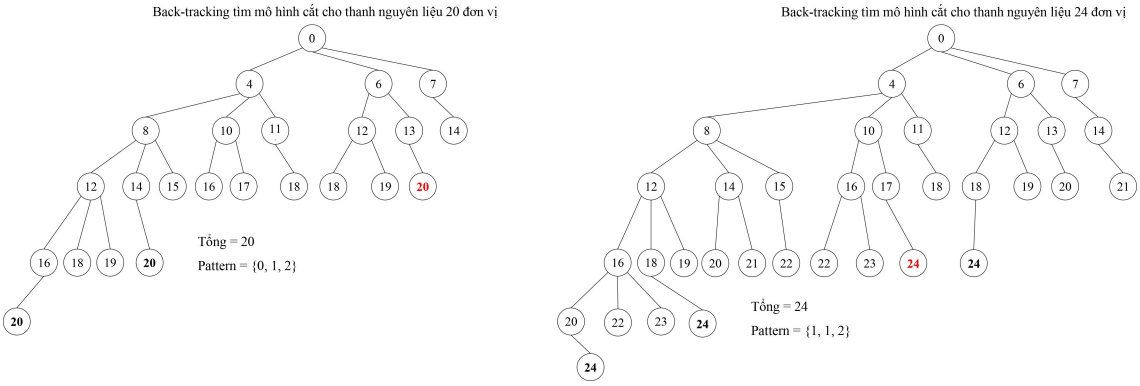
Sử dụng kỹ thuật quay lui (**Backtracking technique**) để tìm cách tận dụng tối đa mỗi loại nguyên liệu. Mô hình cắt **Pattern** được thiết lập có độ dài ban đầu bằng 0, bằng cách thử thêm mỗi mục của đơn hàng vào mô hình cắt hiện tại nếu nó không vượt quá độ dài nguyên liệu thô, sau đó thử bằng một mục khác để kiểm tra các khả năng khác. Mô hình cắt tiết kiệm nhất là mô hình có tổng độ dài gần nhất với nguyên liệu thô. Các phần tử thuộc vector **Pattern** sẽ không bao giờ có giá trị lớn hơn các phần tử thuộc vector **Demand** tương ứng, đảm bảo số không vượt quá yêu cầu đơn hàng.
Dưới đây là minh họa cho quá trình quay lui để tìm ra mô hình cắt tối ưu nhất cho thanh nguyên liệu thô dài 15 đơn vị trong ví dụ:




Với kỹ thuật quay lui, chúng ta nhận được kết quả là mẫu cắt có tổng chiều dài 15, vector **Pattern** = { 2, 0, 1 } mang ý nghĩa mô hình cắt bao gồm 2 mẫu dài 4 đơn vị, 0 mãu dài 6 đơn vị và 1 mẫu dài 7 đơn vị.

Trường hợp thu được nhiều hơn 1 mô hình cắt tối ưu là một vấn đề lớn khiến cho giải thuật này khó đưa đáp án đúng nhất khi lượng dữ liệu đầu vào lớn và khó kiểm soát. Một cách giải quyết là tiếp tục áp dụng quay lui cho bước tiếp theo, bằng cách thử áp dụng một mô hình cắt đã tìm được sau đó quay lui về lúc chưa tiến hành áp dụng mô hình đó để thử áp dụng mô hình cắt tiếp theo. Tuy nhiên cách giải quyết này có độ phức tạp về thời gian là rất lớn đồng thời nảy sinh thêm nhiều vấn đề đi kèm khi hiện thực giải thuật (đặc biệt là rủi ro bị tràn dữ liệu) nên chúng ta không chọn hướng đi này. Mô hình cắt gồm nhiều loại mẫu khác nhau nhất sẽ được chọn làm tiêu chí ưu tiên, nếu số loại mẫu sử dụng là bằng nhau, thì sẽ chọn mô hình cắt sử dụng ít mẫu vật hơn, nếu tổng số mẫu vật đều bằng nhau thì sẽ chọn mô hình cắt tìm thấy trước.

Trở lại ví dụ, áp dụng quay lui để tìm kiếm mô hình cắt phù hợp nhất với mỗi thanh nguyên liệu 20 và 24 đơn vị độ dài, ta thu được kết quả sau:
- Đối với thanh 20 đơn vị, chúng ta thu được 3 mô hình cắt tối ưu nhất là { 5, 0, 0 }, { 2, 2, 0 } và { 0, 1, 2 }. Mô hình cắt đầu tiên không được chọn do chỉ sử dụng 1 loại mẫu trong khi hai mô hình còn lại đều sử dụng 2 mẫu khác nhau. Trong hai mô hình còn lại, mô hình thứ hai sử dụng nhiều mẫu hơn nên mô hình cắt thứ ba được chọn làm mô hình cắt tối ưu nhất.
- Đối với thanh 24 đơn vị, chúng ta thu được 4 mô hình cắt tối ưu là { 6, 0, 0 }, { 3, 2, 0 }, { 0, 4, 0 } và { 1, 1, 2 }. Mô hình cắt thứ nhất và thứ ba không được chọn do chỉ sử dụng 1 loại mẫu. Trong hai mô hình còn lại, mô hình thứ hai sử dụng nhiều mẫu cắt hơn mô hình cuối nên mô hình cuối cùng được chọn làm mô hình cắt tối ưu nhất.




Tiếp theo, cần chọn nguyên liệu có mô hình cắt phù hợp nhất trong việc đáp ứng nhu cầu của đơn hàng, mô hình này sẽ được áp dụng cho đến khi không thể áp dụng được nữa (**nguyên lý tham lam**), sau đó đơn hàng (vector **Demand**) sẽ được cập nhật lại. Chúng ta sẽ thiết lập lại mô hình cắt tối ưu nhất cho từng thanh nguyên liệu, tiếp tục chọn thanh nguyên liệu cho ra mô hình tối ưu nhất để áp dụng. Lặp lại quá trình này cho đến khi đáp ứng xong toàn bộ đơn hàng (toàn bộ phần tử của vector **Demand** đều bằng 0). Các tiêu chí để chọn ra nguyên liệu sẽ phụ thuộc vào mục đích của bài toán là làm giảm tối đa hao phí nguyên liệu hay tiết kiệm tối đa chi phí bỏ ra để mua nguyên liệu và cắt.

Với mục đích **làm giảm tối đa hao phí nguyên liệu**, thanh nguyên liệu được chọn là thanh nguyên liệu có chênh lệch độ dài với mô hình cắt của nó một giá trị nhỏ nhất (mô hình cắt khi áp dụng sẽ ít hao phí nhất). Trường hợp có nhiều thanh nguyên liệu cùng thỏa mãn tiêu chí trên, ta chọn nguyên liệu có độ dài lớn hơn để áp dụng.

Xét đến ví dụ, các thanh nguyên liệu 15, 20, 24 đơn vị có chênh lệch độ dài với mô hình cắt của chúng đều bằng 0, thanh 24 đơn vị sẽ được chọn để áp dụng. Với vector **Demand** hiện tại là { 90, 50, 100 } và **Pattern** sử dụng là { 1, 1, 2 }, chúng ta sẽ áp dụng được min(90/1, 70/1, 110/2) = 55 lần trước khi chọn một mô hình cắt mới.

Thực hiện cập nhật lại yêu cầu đơn hàng: **Demand** = { 90 - 55 x 1, 70 - 55 x 1, 110 - 2 x 55 } = { 35, 15, 0 }. Tiếp tục tìm ra các mô hình cắt phù hợp nhất cho mỗi thanh nguyên liệu. Ta thu được lần lượt ba **Pattern** { 2, 1, 0 }, { 2, 2, 0 } và { 3, 2, 0 }. Thanh nguyên liệu có mô hình cắt ít lãng phí nhất và dài nhất là thanh 24 đơn vị, chúng ta áp dụng mô hình cắt của nó cho vòng lặp lần này. Số lần áp dụng được là min(35/3, 15/2) = 7.

Tiếp tục cập nhật nhật lại đơn hàng: **Demand** = { 35 - 7 x 3, 15 - 7 x 2, 0 - 7 x 0 } = {

14, 1, 0 }. Từ vector **Demand** mới, tìm ra các mô hình cắt phù hợp nhất của mỗi thanh nguyên liệu. Ta thu được lần lượt ba **Pattern** { 2, 1, 0 }, { 5, 0, 0 } và { 6, 0, 0 }. Thanh nguyên liệu ít lãng phí nhất và dài nhất là thanh 24 đơn vị, mô hình cắt của nó sẽ được áp dụng cho vòng lặp lần này. Số lần áp dụng được là 14/6 = 2.

Tiếp tục thực hiện cập nhật nhật lại yêu cầu đơn hàng: **Demand** = { 14 - 2 x 6, 1 - 2 x 0, 0 - 2 x 0 } = { 2, 1, 0 }. Chúng ta sẽ sử dụng thanh 15 đơn vị cho yêu cầu còn lại. Tổng hao phí sẽ là 1 và số thanh nguyên liệu sử dụng là 55 + 7 + 2 + 1 = 65 thanh.

Với mục đích **tối ưu chi phí**, tiêu chí chọn thanh nguyên liệu sẽ phức tạp hơn. Giải thuật sẽ tiến hành dự đoán chi phí bỏ ra và số mẫu vật còn lại của đơn hàng mỗi lần áp dụng tới khi không áp dụng được nữa mô hình cắt của các thanh nguyên liệu. Mô hình cắt được chọn sẽ là mô hình có tích của **chi phí bỏ ra** nhân với **trung bình độ dài các mẫu còn lại** (tạm gọi là tích dự đoán, chỉ lấy phần nguyên độ dài trung bình) thấp nhất. Sau đó tiến hành cập nhật lại yêu cầu đơn hàng, thiết lập lại mô hình cắt, tiếp tục dự đoán và chọn mô hình cắt phù hợp nhất. Lặp lại quá trình này cho đến khi hoàn thành đơn hàng.

Xét ví dụ, giả sử giá nhập các thanh nguyên liệu lần lượt là 10$, 14$ và 18$. Với các mô hình cắt tối ưu lần lượt là {2, 0, 1}, {0, 1, 2} và {1, 1, 2}. Tổng số mẫu hiện tại của đơn hàng là 90 + 70 + 110 = 270. Giải thuật sẽ dự đoán như sau:

- Thanh nguyên liệu 15 đơn vị sẽ áp dụng được min(90/2, 110/1) = 45 lần. Độ dài trung bình của đơn hàng là $\frac{(90-2\times45)\times4+(70-0\times45)\times6+(110-1\times45)\times7}{270-45\times(2+0+1)} = \frac{0\times4+70\times6+65\times7}{135} = 6$. Tích dự đoán là 10 x 45 x 6 = 2700.
- Thanh nguyên liệu 20 đơn vị sẽ áp dụng được min(70/1, 110/2) = 55 lần. Độ dài trung bình của đơn hàng là $\frac{(90-0\times55)\times4+(70-1\times55)\times6+(110-2\times55)\times7}{270-55\times(0+1+2)} = \frac{90\times4+15\times6+0\times7}{105} = 4$. Tích dự đoán là 14 x 55 x 4 = 3080.
- Thanh nguyên liệu 24 đơn vị sẽ áp dụng được min(90/1, 70/1, 110/2) = 55 lần. Độ dài trung bình của đơn hàng là $\frac{(90-1\times55)\times4+(70-1\times55)\times6+(110-2\times55)\times7}{270-55\times(1+1+2)} = \frac{35\times4+15\times6+0\times7}{50} = 4$. Tích dự đoán là 18 x 55 x 4 = 3960.

Tích dự đoán nhỏ nhất thuộc về thanh nguyên liệu 15, áp dụng mô hình cắt của nó và cập nhật lại đơn hàng **Demand** = {0, 70, 65}. Chi phí bỏ ra là 45 x 10 = 450$.
Tiến hành thiết lập lại mô hình cắt cho mỗi thanh nguyên liệu, lần lượt ta có {0, 0, 2}, {0, 1, 2} và {0, 4, 0}. Thực hiện dự đoán:

- Thanh 15 đơn vị sẽ áp dụng được 65/2 = 32 lần. Độ dài trung bình của đơn hàng là $\frac{0\times4+(70-0\times32)\times6+(65-2\times32)\times7}{135-32\times(0+0+2)} = \frac{0+70\times6+1\times7}{71} = 6$. Tích dự đoán là 10 x 32 x 6 = 1920.
- Thanh nguyên liệu 20 đơn vị sẽ áp dụng được min(70/1, 65/2) = 32 lần. Độ dài trung bình của đơn hàng là $\frac{0\times4+(70-1\times32)\times6+(65-2\times32)\times7}{135-32\times(0+1+2)} = \frac{0+38\times6+1\times7}{39} = 6$. Tích dự đoán là 14 x 32 x 6 = 2688
- Thanh nguyên liệu 24 đơn vị sẽ áp dụng được 70/4 = 17 lần. Độ dài trung bình của đơn hàng là $\frac{0\times4+(70-4\times17)\times6+(65-0\times17)\times7}{135-17\times(0+4+0)} = \frac{0+2\times6+65\times7}{67} = 6$. Tích dự đoán là 18 x 17 x 6 = 1836.

Tích dự đoán nhỏ nhất thuộc về thanh nguyên liệu 24, áp dụng mô hình cắt của nó và cập nhật lại đơn hàng **Demand** = {0, 2, 65}. Chi phí bỏ ra là 17 x 18 = 306$.
Lại tiến hành thiết lập mô hình cắt mới cho mỗi thanh nguyên liệu, ta có {0, 0, 2}, {0, 1, 2} và {0, 0, 3}. Thực hiện dự đoán:

- Thanh 15 đơn vị áp dụng được 65/2 = 32 lần. Độ dài trung bình của đơn hàng là

$\frac{0+2\times6+1\times7}{3} = 6$. Tích dự đoán là 10 x 32 x 6 = 1920.
- Thanh nguyên liệu 20 đơn vị áp dụng được min(2/1, 65/2) = 2 lần. Độ dài trung bình của đơn hàng là $\frac{0+0\times6+61\times7}{61} = 7$. Tích dự đoán là 14 x 2 x 7 = 196
- Thanh nguyên liệu 24 đơn vị áp dụng được 65/3 = 21 lần. Độ dài trung bình của đơn hàng là $\frac{0+2\times6+2\times7}{4} = 6$. Tích dự đoán là 18 x 21 x 6 = 2268.

Tích dự đoán nhỏ nhất thuộc về thanh nguyên liệu 20, áp dụng mô hình cắt của nó và cập nhật lại đơn hàng **Demand** = {0, 0, 61}. Chi phí bỏ ra là 2 x 14 = 28$.
Với lượng yêu cầu còn lại, chúng ta sẽ sử dụng thanh 15 đơn vị 31 lần. Chi phí bỏ ra là 31 x 10 = 310$.

Như vậy, thuật toán đưa ra giải pháp sử dụng tổng 45 + 17 + 2 + 31 = 95 thanh nguyên liệu với tổng chi phí là 450 + 306 + 28 + 310 = 1094$.

Giải thuật của thuật toán này khi giải quyết vấn đề tối ưu chi phí chưa thực sự tối ưu trong tất cả các trường hợp. Tuy nhiên, nhóm tác giả cho rằng hướng đi của thuật toán khá ổn khi kiểm tra các trường hợp thực tế.

## 4.3 Phương pháp suy nghiệm sử dụng kĩ thuật xây dựng (Modified FFD Heuristic)

### 4.3.1 Mô hình hóa

Một cách để xác định nghiệm nguyên cho bài toán one dimensional cutting bao gồm việc xây dựng một mẫu cắt tốt và sử dụng nó nhiều lần nhất có thể, giảm bớt sự dư thừa trong sản xuất. Tại mỗi lần lặp lại của thuật toán, nhu cầu của các mặt hàng được cập nhật và quá trình này được lặp lại cho đến khi nhu cầu được cung cấp đầy đủ, ta được nghiệm nguyên của bài toán.

### 4.3.2 Phân tích thuật toán

Thuật toán bắt đầu bằng việc sắp xếp các thanh sản phẩm theo thứ tự giảm dần cùng với nhu cầu của từng thanh. Sau đó sử dụng vòng lặp để chèn từng thanh sản phẩm vào thanh nguyên liệu theo thứ tự giảm dần độ dài, cuối cùng lựa chọn mẫu cắt có lượng dư thừa bé nhất. Cố gắng tận dụng mẫu cắt đó cho đến khi nhu cầu còn lại của một thanh bất kì không thỏa mãn, lặp lại thuật toán khi nhu cầu còn lại của tất cả thanh bằng 0.

Dưới đây là mã giả của giải thuật FFD gốc mà chúng tôi đã chỉnh sửa:

**FFD Heuristic**

**BEGIN**

**P.1.** Sort the items in decreasing order of size.

Suppose $l_1 \geq l_2 \geq \cdots \geq l_m$

**P.2.** Be $r_i$ the residual demand of the item $i \in I$.

$I = \{1, \ldots, m\}$. {set of indexes of the items}

At first: $r_i = d_i, \forall i \in I$.

Do $k = 1$ {First cutting pattern }

STOP=False {logic variable tht indicates a non-null demand}

**While** STOP=False

**P.3.** Do: $Rest = L$ and $\alpha_{ik} = 0, \forall i \in I$;

Be $i = 1$ {start by putting the first item in the pattern }

**While** ($i \leq m$ and $Rest \geq l_i$) Do:

$\alpha_{ik} = \min \left\{ \left\lfloor \frac{Rest}{l_i} \right\rfloor, r_i \right\}$

($\alpha_{ik}$ is the quantity of items type $i$ in pattern $k$)

Do: $Rest = Rest - (\alpha_{ik} l_i)$

$r_i = r_i - \alpha_{ik}$; $i = i + 1$

**End of While**

**P.4.** Determine the frequency of pattern $k$: $x_k = \left\{ \begin{array}{l} \min \left\lfloor \frac{d_i}{\alpha_{ik}} \right\rfloor, \\ \forall i \in I; \alpha_{ik} > 0 \end{array} \right\}$

**P.5.** (Stop Criterion)

**If** $r_i = 0, \ \forall i \in I$ **then** STOP = Truth.

**Otherwise** Do $k = k + 1$ and come back to **P.3**

**End of While.**

**END**

Chỉnh sửa:

- Ở khâu chọn mẫu cắt, thay vì kết thúc vòng lặp khi lượng nguyên liệu dư nhỏ hơn độ dài thanh sản phẩm, chúng tôi tiếp tục duyệt qua những độ dài nhỏ hơn để tận dụng tối đa thanh nguyên liệu.
- Đối với trường hợp có nhiều thanh nguyên liệu, chúng tôi duyệt lần lượt qua từng thanh và chọn mẫu cắt từ thanh có phần dư nguyên liệu bé nhất.
- Số lượng cần cắt của mỗi mẫu là số lượng bé nhất để lấp đầy nhu cầu còn lại của một

thanh sản phẩm bất kì. Cập nhật lại nhu cầu của các thanh sau khi đã chọn mẫu cắt và tiếp tục vòng lặp.

# 5 Kiểm thử và đánh giá

## 5.1 Giải thuật simplex

Kết quả chạy giải thuật với tập dữ liệu kiểm thử:

```
Total time (CPU seconds):       0.01   (Wallclock seconds):       0.01

Status: Optimal
Stock length: [15]
Stock cost: [20]
Product length: [4, 6, 7]
Product demand: [80, 50, 100]

Pattern usage:
Pattern [2, 0, 1] used 40.0 times
Pattern [0, 2, 0] used 25.0 times
Pattern [0, 0, 2] used 30.0 times
Total used: 95.0
Total cost: 1900.0
Total waste: 105.0
```

```
Total time (CPU seconds):       7.30   (Wallclock seconds):       7.30

Status: Optimal
Stock length: [100, 90, 80, 75, 50, 120]
Stock cost: [25, 20, 18, 17, 12, 30]
Product length: [45, 36, 30, 15, 40, 54, 10, 20, 24, 12, 18, 25, 49]
Product demand: [100, 200, 500, 150, 250, 350, 100, 200, 300, 500, 200, 200, 150]

Pattern usage:
Pattern [2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] used 50.0 times
Pattern [0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0] used 163.0 times
Pattern [0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0] used 37.0 times
Pattern [0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] used 117.0 times
Pattern [0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0] used 149.0 times
Pattern [0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0] used 1.0 times
Pattern [0, 0, 0, 0, 2, 0, 1, 0, 0, 0, 0, 0, 0] used 45.0 times
Pattern [0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0] used 5.0 times
Pattern [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 2, 0] used 2.0 times
Pattern [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1] used 150.0 times
Pattern [0, 0, 0, 0, 0, 1, 0, 0, 0, 3, 0, 0, 0] used 152.0 times
Pattern [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2, 0, 0] used 35.0 times
Pattern [0, 0, 0, 0, 0, 0, 0, 9, 0, 0, 0, 0, 0] used 6.0 times
Pattern [0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0] used 23.0 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 1, 0, 0] used 87.0 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 1, 0, 0] used 1.0 times
Pattern [0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0] used 1.0 times
Total used: 1024.0
Total cost: 20478.0
Total waste: 150.0
```

**Nhận xét:**

- Hiệu quả: Vì nghiệm của thuật toán Simplex là nghiệm tối ưu nhất nên kết quả sẽ mang tính chính xác vô cùng cao và phù hợp với các tập dữ liệu yêu cầu chặt chẽ về mặt chi phí.

- Tính ổn định: Simplex là một phương pháp đã được chứng minh là hoạt động tốt trong thực tế với nhiều bài toán LP có kích thước vừa và nhỏ.

**Nhược điểm:**

- Đối với các bài toán có yêu cầu về đa dạng kích thước sản phẩm với số lượng lớn, thuật toán back-tracking sử dụng để tạo các mẫu cắt có thể đạt độ phức tạp thời gian rất lớn. Việc duyệt lặp đi lặp lại tập mẫu cắt để tìm kiếm nghiệm tối ưu tiêu tốn nhiều tài nguyên thời gian nhưng mang lại tối ưu chi phí không đáng kể đối với tập mẫu có sự đồng đều về mặt chi phí nguyên liệu đầu vào.
- Tính toán số nguyên đối với bài toán CSP yêu cầu các biến quyết định là số nguyên, nhưng Simplex chỉ giải quyết được bài toán LP với biến liên tục. Do đó, cần phải sử dụng các kỹ thuật bổ sung như branch-and-bound để tìm nghiệm nguyên.

## 5.2 Phương pháp suy nghiệm với back-tracking

Kết quả chạy giải thuật với tập dữ liệu kiểm thử:

```
Stock length: [15]
Stock cost: [20]
Product length: [4, 6, 7]
Product demand: [80, 50, 100]

Authors's Heuristics Approach
Pattern usage:
Pattern [0, 0, 2] uses 50 times
Pattern [2, 1, 0] uses 40 times
Pattern [0, 2, 0] uses 5 times
Total stock used: 95
Total cost: 1900
Total waste: 105
Run time: 0.004s
```

```
Stock length: [100, 90, 80, 75, 50, 120, 150, 70, 60, 50, 55, 65, 30, 85, 95, 105, 110, 40, 45, 35]
Stock cost: [100, 90, 80, 75, 50, 120, 150, 70, 60, 50, 55, 65, 30, 85, 95, 105, 110, 40, 45, 35]
Product length: [45, 36, 30, 15, 40, 54, 10, 20, 24, 12, 18, 25, 49, 5, 7, 23, 13, 11, 14, 28, 42, 32, 35, 22]
Product demand: [150, 300, 200, 250, 300, 400, 300, 150, 200, 250, 175, 180, 240, 320, 225, 300, 175, 280, 320, 120, 75, 80, 220, 160]

Authors's Heuristics Approach
Pattern usage:
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3] uses 53 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1] uses 1 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0, 0, 3, 0, 0] uses 26 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0] uses 2 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0, 0] uses 119 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0] uses 1 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 2, 0, 0, 0, 0] uses 65 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0] uses 1 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0] uses 57 times
Pattern [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0] uses 1 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 1, 0, 0, 0, 0, 0] uses 52 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0] uses 1 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 3, 1, 0, 0] uses 40 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0] uses 17 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0] uses 1 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0] uses 207 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 2, 0] uses 10 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] uses 175 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] uses 74 times
Pattern [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] uses 106 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] uses 50 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] uses 150 times
Pattern [0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] uses 96 times
Pattern [0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] uses 1 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0] uses 199 times
Pattern [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] uses 200 times
Pattern [0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] uses 124 times
Pattern [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] uses 1 times
Pattern [0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] uses 100 times
Pattern [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] uses 240 times
Pattern [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] uses 150 times
Pattern [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] uses 300 times
Pattern [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] uses 400 times
Total stock used: 3020
Total cost: 140730
Total waste: 6420
Run time: 24.797s
```

**Nhận xét:**

- Hiệu quả: Thuật toán giải quyết được các trường hợp một hay nhiều thanh nguyên liệu mẫu với khả năng tối ưu tương đối cao.
- Tính ổn định: Vì không cần thao tác trên ma trận nên thuật toán có thể xử lí các tập dữ liệu lớn hơn và trong thời gian nhanh hơn so với thuật toán Simplex truyền thống.

**Nhược điểm:**

- Vì sử dụng giải thuật back-tracking nên việc tìm ra mẫu cắt tối ưu vẫn tiêu tốn khá nhiều thời gian.
- Khả năng tối ưu chi phí cũng như tối ưu phần dư nguyên liệu sẽ không phải là phương án hiệu quả nhất khi so với giải thuật truyền thống.

## 5.3 Phương pháp suy nghiệm với giải thuật FFD

Kết quả chạy giải thuật với tập dữ liệu kiểm thử:

```
Stock length: [15]
Stock cost: [20]
Product length: [4, 7, 6]
Product demand: [80, 100, 50]

Patterns will be printed in decreasing order of length !
Usage pattern:
Used 50 times from stock length 15:[2, 0, 0]
Used 25 times from stock length 15:[0, 2, 0]
Used 26 times from stock length 15:[0, 0, 3]
Used 1 times from stock length 15:[0, 0, 2]
Total waste: 210
Total cost: 2040
Run time: 0.005s
```

```
Stock length: [100, 90, 80, 75, 50, 120, 150, 70, 60, 50, 55, 65, 30, 85, 95, 105, 110, 40, 45, 35]
Stock cost: [100, 90, 80, 75, 50, 120, 150, 70, 60, 50, 55, 65, 30, 85, 95, 105, 110, 40, 45, 35]
Product length: [45, 36, 30, 15, 40, 54, 10, 20, 24, 12, 18, 25, 49, 5, 7, 23, 13, 11, 14, 28, 42, 32, 35, 22]
Product demand: [150, 300, 200, 250, 300, 400, 300, 150, 200, 250, 175, 180, 240, 320, 225, 300, 175, 280, 320, 120, 75, 80, 220, 160]

Patterns will be printed in decreasing order of length !
Usage pattern:
Used 300 times from stock length 90:[1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Used 50 times from stock length 120:[2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
Used 120 times from stock length 120:[0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
Used 75 times from stock length 100:[0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
Used 25 times from stock length 150:[0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Used 150 times from stock length 100:[0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
Used 110 times from stock length 100:[0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Used 40 times from stock length 75:[0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
Used 30 times from stock length 100:[0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
Used 60 times from stock length 80:[0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Used 45 times from stock length 100:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Used 38 times from stock length 90:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
Used 1 times from stock length 100:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0, 0, 0, 0, 0, 0, 0, 1, 0]
Used 99 times from stock length 80:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 1, 0, 0]
Used 10 times from stock length 100:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 1, 0, 0]
Used 27 times from stock length 100:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 1, 0]
Used 1 times from stock length 80:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 1, 0, 0, 0, 0, 0]
Used 41 times from stock length 100:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 1, 0]
Used 1 times from stock length 100:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 5, 0, 0, 0, 0]
Used 62 times from stock length 80:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 1, 0, 0]
Used 1 times from stock length 100:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 3, 0, 0, 0, 1]
Used 28 times from stock length 90:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 1, 0, 0, 0]
Used 1 times from stock length 100:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 4, 0, 0, 0]
Used 15 times from stock length 120:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 0, 0, 0]
Used 1 times from stock length 150:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 4, 1, 0, 0]
Used 13 times from stock length 120:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 1, 0, 0]
Used 1 times from stock length 60:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 1]
Used 1 times from stock length 100:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 7, 1, 0]
Used 4 times from stock length 100:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 0, 0]
Used 1 times from stock length 80:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 5, 1]
Used 21 times from stock length 75:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 1]
Used 1 times from stock length 40:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 1]
Used 1 times from stock length 100:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 15]
Used 14 times from stock length 100:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 20]
Total waste: 5
Total cost: 134315
Run time: 0.108s
```

**Nhận xét:**

- Hiệu quả: Thuật toán giải quyết được các trường hợp một hay nhiều thanh nguyên liệu mẫu với khả năng tối ưu tương đối cao.
- Tính ổn định: Thuật toán giản lược bớt các thao tác chọn mẫu có thể xử lí các tập dữ liệu

lớn hơn và trong thời gian nhanh hơn so với thuật toán Simplex truyền thống cũng như giải thuật suy nghiệm back-tracking.

- Thuật toán đặc biệt hiệu quả trên tập dữ liệu thực tế khi mà chi phí tỉ lệ thuận gần tuyến tính với độ dài thanh nguyên liệu.

**Nhược điểm:**

- Thuật toán ưu tiên xử lí các thanh sản phẩm có độ dài theo thứ tự giảm dần nên sẽ kém hiệu quả đối với dữ liệu có độ lớn thah sản phẩm quá đồng đều.
- Khả năng tối ưu chi phí cũng như tối ưu phần dư nguyên liệu sẽ không phải là phương án hiệu quả nhất khi so với giải thuật truyền thống.

## 5.4 Kiểm tra với tập dữ liệu từ bài toán thực tế bằng công cụ Excel

### 5.4.1 Đầu vào

Để có thể sử dụng công cụ solver, ta cần cung cấp thông tin các điều kiện ràng buộc và hàm mục tiêu. Thông tin về các điều kiện được cung cấp qua các bảng thông tin sau:

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | Bảng phân loại và giá thu mua đối tác cung cấp gỗ | | | | | | | |
| 2 | Phân loại | | | | | Chiều dài (mét) | | Giá thu mua (triệu đồng) |
| 3 | Cây dưới 15 năm tuổi (Phân loại M) | | | | | 16 | | 9 |
| 4 | Cây từ 15 đến dưới 20 năm tuổi (Phân loại L) | | | | | 20 | | 13 |
| 5 | Cây 20 năm tuổi trở lên (Phân loại SL) | | | | | 27 | | 20 |

Hình 6: Bảng phân loại các nguồn gỗ được sử dụng (Stocks)

| | L | M | N | O | P | Q | R | S |
|---|---|---|---|---|---|---|---|---|
| 1 | Số lượng gỗ cần thiết để sản xuất đơn hàng | | | | | | | |
| 2 | Chiều dài (mét) | | 4 | 5 | 9 | 12 | 13 | 21 |
| 3 | Số lượng | | 28 | 300 | 290 | 150 | 187 | 320 |

Hình 7: Bảng phân loại các chi tiết cần cắt (Finish)

Các mẫu cắt được sinh tự động bằng code nhóm:

| | Bảng chi tiết cách cắt | | | Chi tiết cắt phân loại số lượng cắt theo kích thước yêu cầu | | | | | | Lãng phí 1 lần cắt | Số lượng chi tiết cắt |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | Phân loại | Kích thước | STT | 4 | 5 | 9 | 12 | 13 | 21 | | |
| 9 | M | 16 | 1 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | | | 2 | 2 | 1 | 0 | 0 | 0 | 0 | 3 | 0 |
| 11 | | | 3 | 1 | 2 | 0 | 0 | 0 | 0 | 2 | 0 |
| 12 | | | 4 | 1 | 0 | 1 | 0 | 0 | 0 | 3 | 0 |
| 13 | | | 5 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 14 | | | 6 | 0 | 3 | 0 | 0 | 0 | 0 | 1 | 0 |
| 15 | | | 7 | 0 | 1 | 1 | 0 | 0 | 0 | 2 | 0 |
| 16 | | | 8 | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 0 |

Hình 8: Mẫu các chi tiết cắt từ thân gỗ M

| | Bảng chi tiết cách cắt | | | Chi tiết cắt phân loại số lượng cắt theo kích thước yêu cầu | | | | | | Lãng phí 1 lần cắt | Số lượng chi tiết cắt |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | Phân loại | Kích thước | STT | 4 | 5 | 9 | 12 | 13 | 21 | | |
| 17 | L | 20 | 9 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | | | 10 | 3 | 1 | 0 | 0 | 0 | 0 | 3 | 0 |
| 19 | | | 11 | 2 | 2 | 0 | 0 | 0 | 0 | 2 | 0 |
| 20 | | | 12 | 2 | 0 | 1 | 0 | 0 | 0 | 3 | 0 |
| 21 | | | 13 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 22 | | | 2 | 1 | 3 | 0 | 0 | 0 | 0 | 1 | 0 |
| 23 | | | 3 | 1 | 1 | 1 | 0 | 0 | 0 | 2 | 0 |
| 24 | | | 4 | 1 | 0 | 0 | 0 | 1 | 0 | 3 | 0 |
| 25 | | | 5 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 |
| 26 | | | 6 | 0 | 2 | 1 | 0 | 0 | 0 | 1 | 0 |
| 27 | | | 7 | 0 | 1 | 0 | 1 | 0 | 0 | 3 | 0 |
| 28 | | | 8 | 0 | 1 | 0 | 0 | 1 | 0 | 2 | 0 |
| 29 | | | 9 | 0 | 0 | 2 | 0 | 0 | 0 | 2 | 0 |

Hình 9: Mẫu các chi tiết cắt từ thân gỗ L

| | Bảng chi tiết cách cắt | | | Chi tiết cắt phân loại số lượng cắt theo kích thước yêu cầu | | | | | | Lãng phí 1 lần cắt | Số lượng chi tiết cắt |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | Phân loại | Kích thước | STT | 4 | 5 | 9 | 12 | 13 | 21 | | |
| 30 | SL | 27 | 10 | 6 | 0 | 0 | 0 | 0 | 0 | 3 | 0 |
| 31 | | | 11 | 5 | 1 | 0 | 0 | 0 | 0 | 2 | 0 |
| 32 | | | 12 | 4 | 2 | 0 | 0 | 0 | 0 | 1 | 0 |
| 33 | | | 13 | 4 | 0 | 1 | 0 | 0 | 0 | 2 | 0 |
| 34 | | | 14 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| 35 | | | 15 | 3 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 36 | | | 16 | 3 | 0 | 0 | 1 | 0 | 0 | 3 | 0 |
| 37 | | | 17 | 3 | 0 | 0 | 0 | 1 | 0 | 2 | 0 |
| 38 | | | 18 | 2 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| 39 | | | 19 | 2 | 1 | 0 | 1 | 0 | 0 | 2 | 0 |
| 40 | | | 1 | 2 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 41 | | | 2 | 2 | 0 | 2 | 0 | 0 | 0 | 1 | 0 |
| 42 | | | 3 | 1 | 4 | 0 | 0 | 0 | 0 | 3 | 0 |
| 43 | | | 4 | 1 | 2 | 0 | 1 | 0 | 0 | 1 | 0 |
| 44 | | | 5 | 1 | 2 | 0 | 0 | 1 | 0 | 0 | 0 |
| 45 | | | 6 | 1 | 1 | 2 | 0 | 0 | 0 | 0 | 0 |
| 46 | | | 7 | 1 | 0 | 1 | 1 | 0 | 0 | 2 | 0 |
| 47 | | | 8 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 48 | | | 9 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 0 |
| 49 | | | 10 | 0 | 5 | 0 | 0 | 0 | 0 | 2 | 0 |
| 50 | | | 11 | 0 | 3 | 1 | 0 | 0 | 0 | 3 | 0 |
| 51 | | | 12 | 0 | 3 | 0 | 1 | 0 | 0 | 0 | 0 |
| 52 | | | 13 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 53 | | | 14 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 54 | | | 15 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 55 | | | 16 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 |
| 56 | | | 17 | 0 | 0 | 0 | 2 | 0 | 0 | 3 | 0 |
| 57 | | | 18 | 0 | 0 | 0 | 1 | 1 | 0 | 2 | 0 |
| 58 | | | 19 | 0 | 0 | 0 | 0 | 2 | 0 | 1 | 0 |

Hình 10: Mẫu các chi tiết cắt từ thân gỗ SL

| Tổng chi tiết 4 sử dụng | Tổng chi tiết 5 sử dụng | Tổng chi tiết 9 sử dụng | Tổng chi tiết 12 sử dụng | Tổng chi tiết 13 sử dụng | Tổng chi tiết 21 sử dụng |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |

| Tổng số thân M | Tổng số thân L | Tổng số thân SL |
|---|---|---|
| 0 | 0 | 0 |

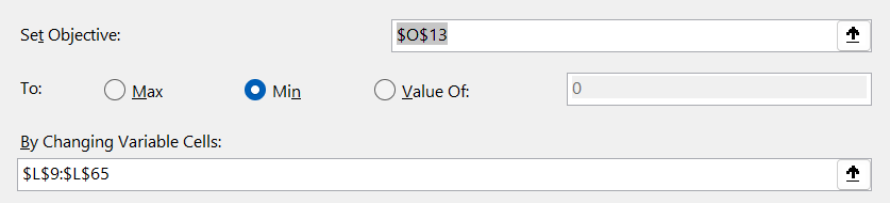
| | | |
|---|---|---|
| Giá tiền: | 0 | triệu đồng |
| Lượng dư thừa - lãng phí: | 0 | mét |

Hình 11: Bảng thông tin chi tiết về số thành phẩm cắt, số thân cây sử dụng và tổng chi phí

**Mô tả rảng buộc:** Như đã đề cập ở **phần 2.3**, bài toán Cutting Stock Problem có điều kiện như sau:

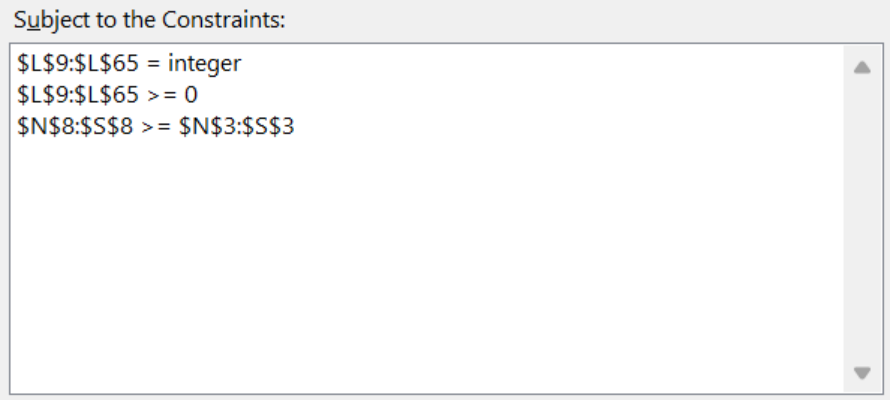
$$\begin{aligned} \min_{x} \quad & \textstyle\sum_{i=1}^{n} \mathbf{c_i x_i} \\ \text{s.t.} \quad & \textstyle\sum_{j=1}^{n} \mathbf{a_{ij} x_j} \ \geq b \, (i = 1, 2, 3, ..., m) \\ & \mathbf{x_i} \geq 0 \ (i = 1, 2, ..., n) \end{aligned} \tag{10}$$

Đây là bài toán cực tiểu với hàm mục tiêu nhằm tối ưu chi phí do đó trong công cụ Solver, ta tiến hành cài đặt giá tiền thành giá trị của hàm mục tiêu và xác định bài toán cực tiểu (minimize) như sau:




Bằng cách thay đổi các giá trị số lượng đã được đề cập qua **hình** 8, **hình** 9 và **hình** 10 mục tiêu nhằm tối ưu hóa chi phí cần thiết.

Với các điều kiện ràng buộc ở trên, ta cần đảm bảo giá trị số lượng là số nguyên không âm và các chi tiết cắt cần phải thỏa mãn số lượng cần thiết, do đó ta tiến hành cài đặt các điều kiện ràng buộc cho Excel Solver như sau:




### 5.4.2 Đầu ra và so sánh

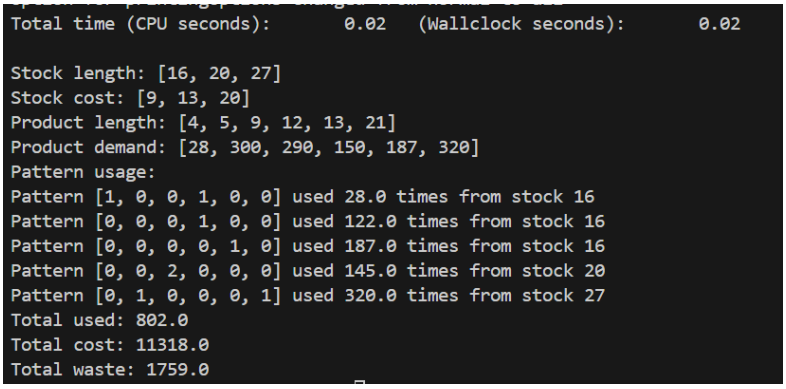
Kết quả thu được khi chạy công cụ Excel Solver và đối chứng kết quả của nhóm như sau:

| Tổng chi tiết 4 sử dụng | Tổng chi tiết 5 sử dụng | Tổng chi tiết 9 sử dụng | Tổng chi tiết 12 sử dụng | Tổng chi tiết 13 sử dụng | Tổng chi tiết 21 sử dụng |
|---|---|---|---|---|---|
| 170 | 300 | 290 | 150 | 187 | 320 |

| Tổng số thân M | Tổng số thân L | Tổng số thân SL |
|---|---|---|
| 337 | 145 | 320 |

| Giá tiền: | 11,318 | triệu đồng |
|---|---|---|

Hình 12: Kết quả chạy Solver




Hình 13: Kết quả chạy thuật toán

Có thể nhận thấy, số lượng nguyên liệu và chi phí là giống nhau, do đó thuật toán mà công cụ Excel Solver sử dụng với thuật toán Simplex là hoàn toàn tương đồng.

# 6 Kết luận

Kết quả xử lí tập dữ liệu đã giới thiệu ở đầu bài viết với 3 thuật toán khác nhau:

Thuật toán Simplex:

```
Total time (CPU seconds):       0.02   (Wallclock seconds):       0.02

Stock length: [16, 20, 27]
Stock cost: [9, 13, 20]
Product length: [4, 5, 9, 12, 13, 21]
Product demand: [28, 300, 290, 150, 187, 320]
Pattern usage:
Pattern [1, 0, 0, 1, 0, 0] used 28.0 times from stock 16
Pattern [0, 0, 0, 1, 0, 0] used 122.0 times from stock 16
Pattern [0, 0, 0, 0, 1, 0] used 187.0 times from stock 16
Pattern [0, 0, 2, 0, 0, 0] used 145.0 times from stock 20
Pattern [0, 1, 0, 0, 0, 1] used 320.0 times from stock 27
Total used: 802.0
Total cost: 11318.0
Total waste: 1759.0
```

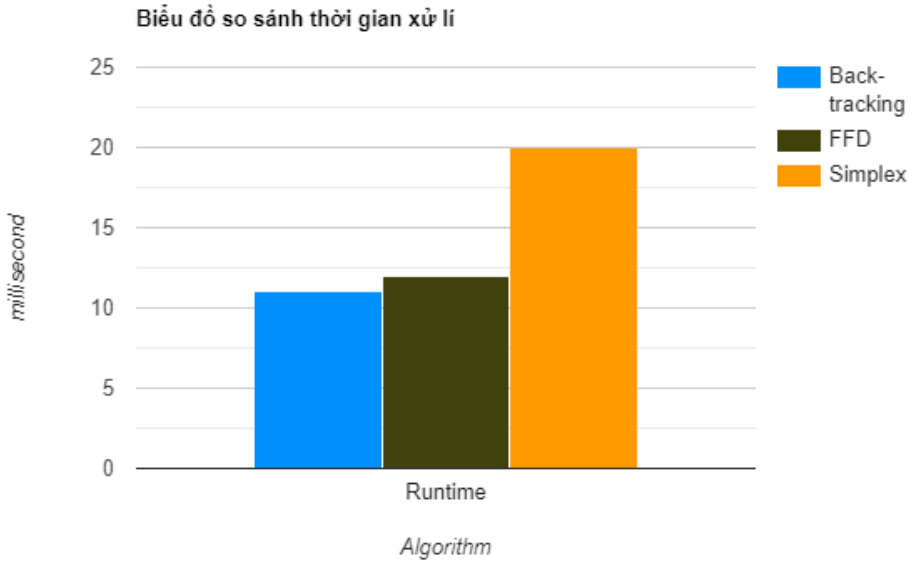
Thuật toán suy nghiệm sử dụng back-tracking:

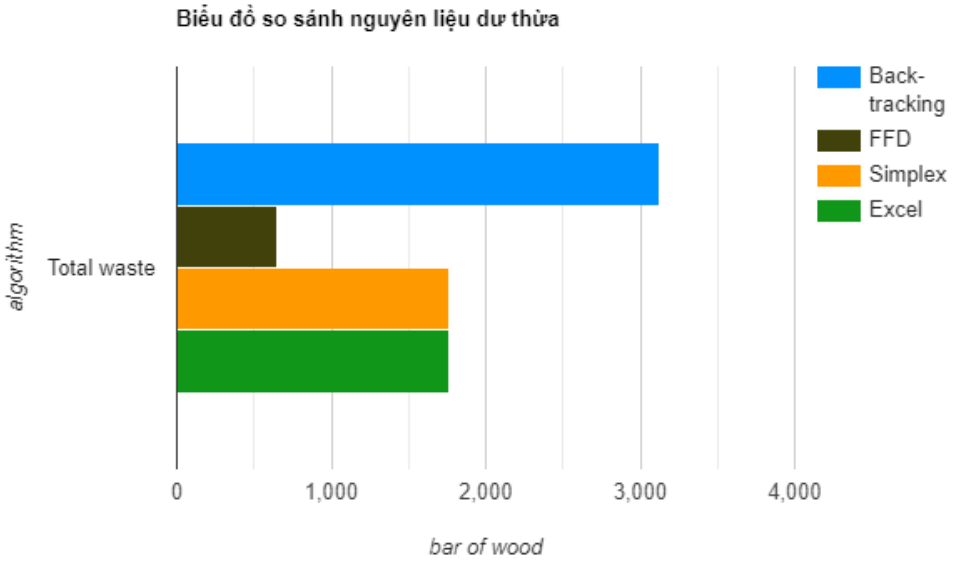
```
Stock length: [16, 20, 27]
Stock cost: [9, 13, 20]
Product length: [4, 5, 9, 12, 13, 21]
Product demand: [28, 300, 290, 150, 187, 320]

Authors's Heuristics Approach
Pattern usage:
Pattern [2, 0, 0, 1, 0, 0] uses 14 times from stock 20
Pattern [0, 3, 0, 0, 0, 0] uses 100 times from stock 16
Pattern [0, 0, 0, 0, 1, 0] uses 187 times from stock 16
Pattern [0, 0, 0, 1, 0, 0] uses 136 times from stock 16
Pattern [0, 0, 3, 0, 0, 0] uses 96 times from stock 27
Pattern [0, 0, 2, 0, 0, 0] uses 1 times from stock 20
Pattern [0, 0, 0, 0, 0, 1] uses 320 times from stock 27
Total stock used: 854
Total cost: 12322
Total waste: 3127
Run time: 0.011s
```

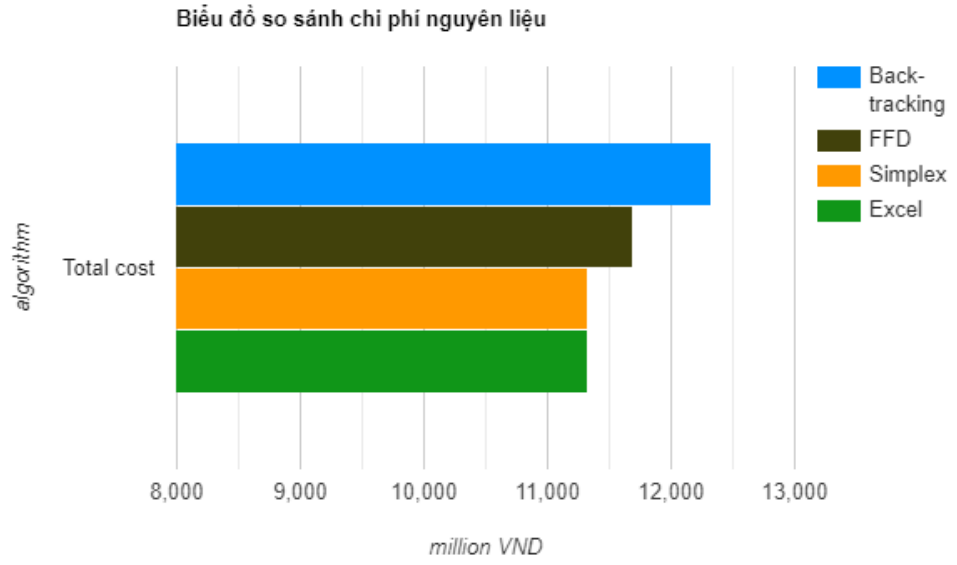
Thuật toán suy nghiệm FFD:

```
Stock length: [16, 20, 27]
Stock cost: [9, 13, 20]
Product length: [4, 5, 9, 12, 13, 21]
Product demand: [28, 300, 290, 150, 187, 320]

Patterns will be printed in decreasing order of length !
Usage pattern:
Used 300 times from stock length 27:[1, 0, 0, 0, 1, 0]
Used 20 times from stock length 27:[1, 0, 0, 0, 0, 1]
Used 93 times from stock length 27:[0, 2, 0, 0, 0, 0]
Used 1 times from stock length 27:[0, 1, 1, 0, 0, 0]
Used 8 times from stock length 16:[0, 0, 1, 0, 0, 1]
Used 70 times from stock length 27:[0, 0, 2, 0, 0, 0]
Used 1 times from stock length 16:[0, 0, 1, 0, 0, 0]
Used 96 times from stock length 27:[0, 0, 0, 3, 0, 0]
Used 1 times from stock length 20:[0, 0, 0, 2, 0, 0]
Total waste: 651
Total cost: 11694
Run time: 0.012s
```

**So sánh kết quả bằng đồ thị:**






**Nhận xét:**

- Đối với tập dữ liệu có kích thước nhỏ, thời gian xử lí của Back-tracking và FFD tương đương nhau và nhanh gấp đôi so với Simplex.
- Khả năng tối ưu phần thừa nguyên liệu của FFD tốt hơn nhiều so với 2 thuật toán còn lại.

- Khả tăng tối ưu chi phí của thuật toán Simplex là tốt nhất, sau đó đến FFD và sau cùng là Back-tracking.

**Kết luận:**

Bài viết đã trình bày và so sánh một số thuật toán để giải quyết bài toán One-dimensional Cutting Stock. Vì Cutting Stock là một bài toán NP-hard nên sẽ không có cách giải quyết tối ưu nhất đối với tập dữ liệu lớn. Đối với thuật toán suy nghiệm, ta sẽ xử lí được các tập dữ liệu phức tạp hơn phù hợp với các doanh nghiệp vừa và lớn, tuy nhiên kết quả sẽ không phải là tối ưu nhất nên cần cân nhắc mức độ ưu tiên giữa các tiêu chí nguyên liệu thừa, tổng chi phí và thời gian thực thi để tìm ra phương pháp tối ưu và phù hợp với bộ dữ liệu. Khi xử lí bộ dữ liệu nhỏ phù hợp với các doanh nghiệp nhỏ, ta có thể sử dụng thuật toán Simplex để được phương án tối ưu nhất nhờ đó tiết kiệm nguyên liệu cũng như chi phí.

Đối với đề xuất cải thiện, khi cần xử lí lượng lớn giữ liệu, ta cần đơn giản bớt các thao tác chọn lựa mẫu cắt nhưng đảm bảo không làm giảm đi quá nhiều hiệu suất nhờ đó đảm bảo thời gian thực thi chương trình trong mức cho phép. Những thuật toán đã được đề xuất hoạt động hiệu quả trên tập dữ liệu thực tế. Trong tương lai, các thuật toán này có thể được cải tiến để phù hợp với các tập dữ liệu khác cũng như giải quyết vấn đề cắt nguyên liệu 2 chiều.

# Tham khảo

[1] Gonçalo Cerqueira, Sérgio Aguiar, and Marlos Marques. Modified greedy heuristic for the one-dimensional cutting stock problem. *Journal of Combinatorial Optimization*, 42:1–18, 10 2021.

[2] Harald Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44:145–159, 01 1990.

[3] Yuval Lirov. Knowledge based approach to the cutting stock problem. *Mathematical and Computer Modelling*, 16(1):107–125, 1992.

[4] Boualem Slimi and Moncef Abbas. Contribution to solving a one-dimensional cutting stock problem with two objectives based on the generation of cutting patterns. *Rect@*, 23:1–22, 2022.

[5] Haihua Xiao, Qiaokang Liang, Dan Zhang, Suhua Xiao, and Gangzhuo Nie. A method for demand-accurate one-dimensional cutting problems with pattern reduction. *Mathematical Biosciences and Engineering*, 20(4):7453–7486, 2023.