

# Preparo de Dados

## 1.1 Dataset

O dataset utilizado está disponível no repositório [FSSP Dataset – GitHub](#), criado por Akil El Kamel.

O conjunto de dados representa um problema clássico de escalonamento de fluxo (Flow Shop Scheduling Problem – FSSP), no qual um conjunto de tarefas deve ser processado em uma sequência fixa de máquinas.

**Contexto:** cada linha representa uma tarefa, e cada coluna corresponde a uma máquina.

**Valores:** o número em cada célula indica o tempo necessário para que uma máquina execute uma determinada tarefa.

**Dimensões:** 10 tarefas (linhas) × 3 máquinas (colunas).

**Objetivo:** minimizar o tempo total de produção (makespan) por meio da melhor sequência de execução das tarefas.

Informações do dataset:

- Colunas relacionadas às máquinas e suas tarefas...
- Com menos dados, fica mais fácil a implementação do algoritmo "Branch and Bound".
- Cada linha representa uma tarefa, e cada coluna representa uma máquina.
- O valor contido em cada célula indica o tempo necessário para a máquina executar a tarefa correspondente."

O dataset FSSP tem algumas variações, optamos por usar o menor dataset com 3 máquinas e 10 jobs.

## 1.2 Dados

Para a verificação de dados foi utilizada **df.info()** e **df.describe()**, implementamos uma abordagem para dados nulos e duplicados:

- **df.info()** - Informações dos tipos de dados -> Inteiros e Object
- **df.isnull().sum()** NÃO TEM FALTA DE DADOS
- **df.duplicated().sum()** NÃO TEM DADOS DUPLICADOS

## Limpeza e Padronização:

- Removendo os dados do tipo Object - coluna '**Unnamed: 0**'.
- Padronização dos nomes das colunas : Maquina\_1, Maquina\_2, Maquina\_3.
- Padronização dos índices (Tarefa\_1, Tarefa\_2, ...).

Padronizando os nomes das colunas e índices para melhorar a legibilidade.

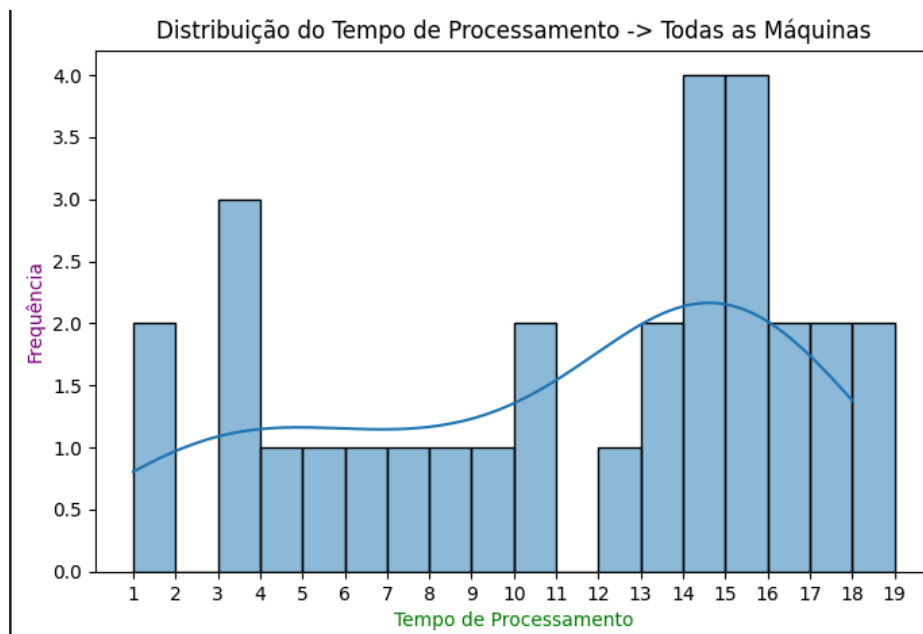
Todos os dados foram tratados, removendo a coluna object, não possui valores nulos no dataset.

**Insights:** Para fazer os gráficos transformamos o data frame para uma lista.

Com base nos gráficos e com a análise podemos tirar algumas conclusões para aplicar o algoritmo Branch and Bound (definindo prioridades e ordem de execução).

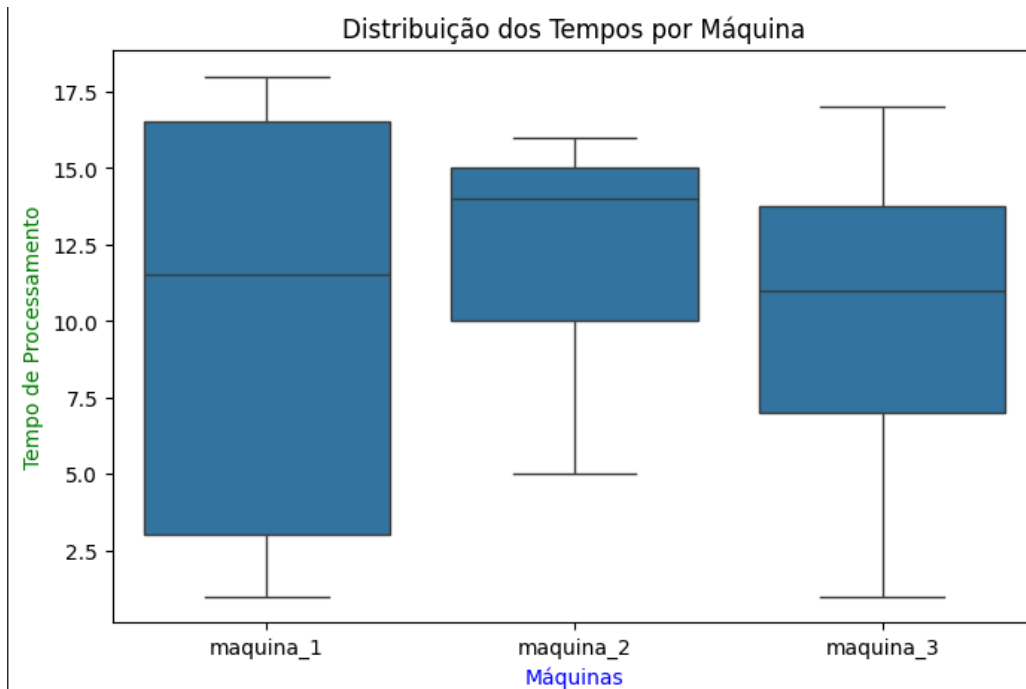
## Gráficos:

### Tempo e frequência:



- Com o gráfico dá para notar que as máquinas passam boa parte do tempo com tarefas.
- Frequência relacionada a quantidade de tarefas...
- As máquinas trabalham com tarefas de complexidades distintas, podendo gerar desequilíbrio.

## Máquinas para determinadas tarefas:



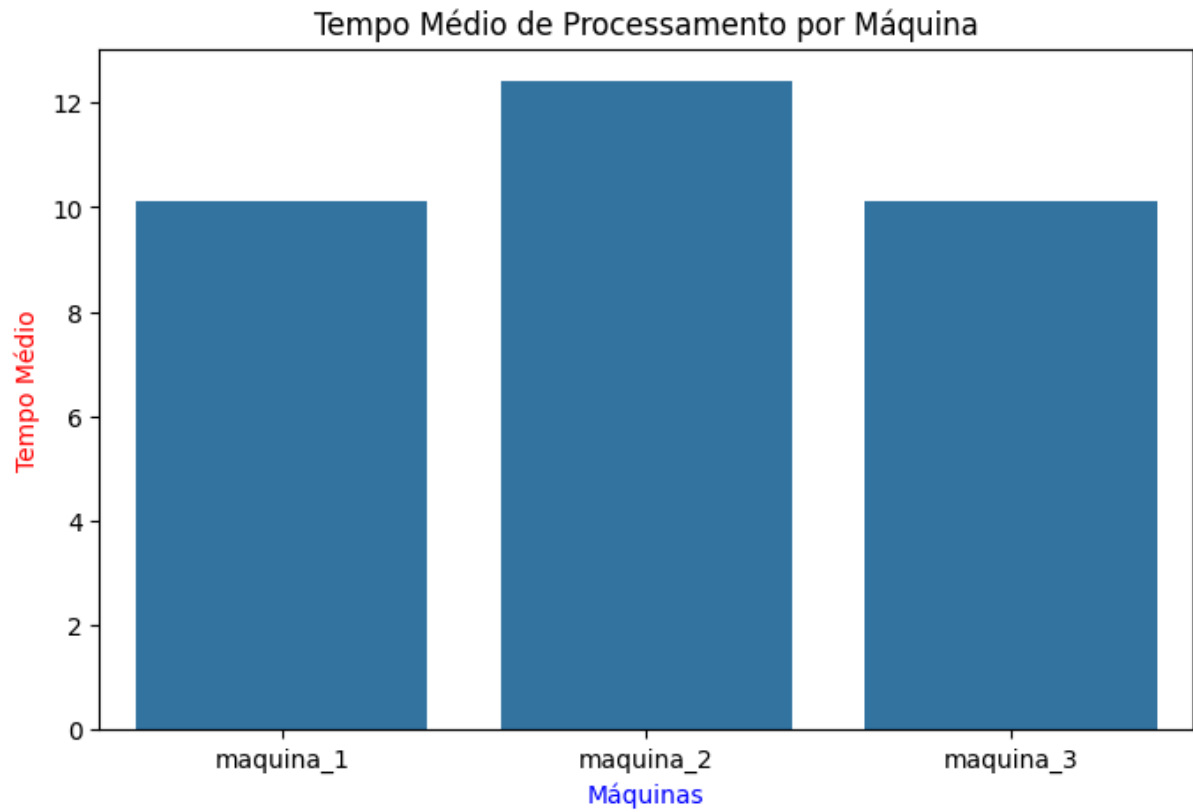
Máquina 1: Apresenta tarefas rápidas e algumas muito demoradas, é bem instável.

Máquina 2: Podemos notar valores parecidos, sem grandes diferenças..

Máquina 3: Comportamento intermediário, não tem valores tão distintos igual a M1. Em relação com as outras máquinas está "equilibrada".

**Valores distintos podem gerar gargalos, causando esperas irregulares na linha de produção.**

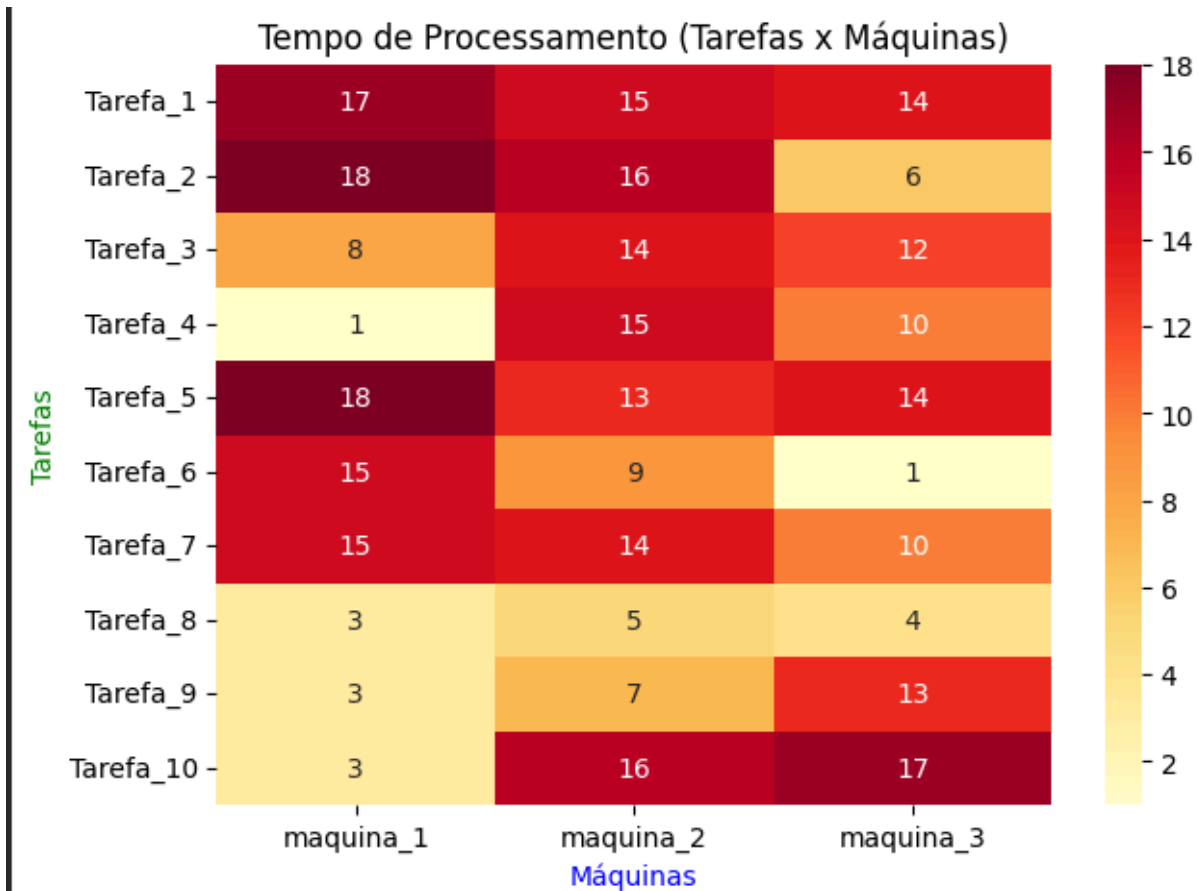
### Média de tempo em cada Máquina:



1 e 3 possuem médias semelhantes, supostamente têm um desempenho equilibrado.

2 possui o maior tempo médio, indício de gargalo...

## HEATMAP:



- Podemos analisar o tempo de cada tarefa com base na cor, por exemplo a máquina 1 tem 2 tarefas com um tempo elevado.
- O tempo de cada tarefa varia de acordo com a máquina. Na tarefa 4 a primeira máquina demorou 1, a segunda máquina foi para 15.
- Para a máquina 2, tem constâncias de tons escuros, reforçando que ela é o ponto de maior carga.
- O heatmap facilita a visualização de gargalos.

## Conclusão:

- Com base nos gráficos, a Máquina 2 tende a ser o gargalo, com maior tempo médio de execução.
- O objetivo é minimizar o tempo total de produção através da melhor sequência de tarefas entre as máquinas.
- Esse tipo de problema é classicamente resolvido por algoritmos como o Branch and Bound.

### 1.3 Definição matemática & Otimização:

- O dataset selecionado representa um conjunto de tarefas (jobs) que devem ser processadas em um conjunto de máquinas.
  - Cada linha corresponde a uma tarefa, e cada coluna representa o tempo de processamento dessa tarefa em uma máquina específica.
  - Este tipo de problema é conhecido como Flow Shop Scheduling Problem (FSSP).
- 
- Tarefas (N): quantidade de jobs que precisam ser processados.
  - Máquinas (M): quantidade de máquinas pelas quais cada tarefa deve passar.

Tempo de processamento:

$\text{tempo\_processamento}[i][j]$  ->  $i$  representa a tarefa e  $j$  representa a máquina.

#### Variáveis de decisão

$S = [s_1, s_2, \dots, s_n]$  -> onde  $S$  é uma permutação das tarefas.

#### Função Objetiva

Minimizar o tempo total de conclusão:

$$\text{Min } C_{\max} = \max_{i,j} (C_{i,j})$$

Onde  $C_{i,j}$  representa o tempo de término da tarefa  $i$  na máquina  $j$ .

#### Restrições:

- Cada tarefa deve passar por todas as máquinas na mesma ordem.
- Cada máquina só pode processar uma tarefa por vez.
- Uma tarefa só pode começar em uma máquina após terminar na máquina anterior.

As restrições garantem a coerência adequada para a técnica Branch and Bound.

#### Makespan:

Implementamos no código uma função para calcular o tempo total (makespan), de uma sequência de tarefas considerando o fluxo em várias máquinas.

## Branch and Bound:

Justificativa para o Branch and Bound

- O espaço de busca contém todas as permutações possíveis das tarefas ( $N!$ ), o que torna inviável uma busca exaustiva para grandes valores de  $N$ .
- O método **Branch and Bound** é adequado porque:
  - Permite explorar o espaço de busca de forma sistemática, avaliando parcialmente as soluções.
  - Usa limites inferiores (bounds) para podar ramos que não podem levar à solução ótima.
  - É amplamente aplicado em problemas de sequenciamento e agendamento (scheduling).

## 1.4 Análise Exploratória

### Estrutura dos Dados:

- Dataset com 10 tarefas e 3 máquinas.
- Sem valores nulos ou inconsistentes.
- Todos os dados numéricos.

### Estatísticas:

Com o **df.describe()** podemos obter a média central de cada máquina.

## Visualização Exploratória:

Foram implementados gráficos do Matplotlib e Seaborn:

- Histograma: Distribuição dos tempos de processamento, as máquinas executam tarefas de complexidade intermediária, com poucas tarefas rápidas e algumas muito lentas.
- Boxplot: A máquina 1 é a mais instável, a 2 apresenta tempos mais altos e consistentes, e a 3 é intermediária.
- Barplot: Confirma que a máquina 2 tem o maior tempo médio, sugerindo ser o ponto crítico do sistema.
- Heatmap: Evidencia que a Máquina 2 possui diversas tarefas com tempos elevados, reforçando o padrão de sobrecarga.

## Interpretação e Hipóteses:

As tarefas possuem tempos desbalanceados entre as máquinas, o que causa esperas irregulares na linha de produção.

Hipótese para modelagem: priorizar tarefas que reduzam o impacto do gargalo na Máquina 2 pode diminuir o makespan total.