UNIVERSITY OF SALENTO

FACULTY OF ENGINEERING

Master's Degree in Management Engineering

Year Project of

PRODUCTION MANAGEMENT AND LEAN MANUFACTURING

OPTIMISING PRODUCTION SCHEDULING IN MATLAB

An advanced approach for industrial order management

Students:
Veronica CRETÌ
Angelo FOGGETTI

ACADEMIC YEAR 2023/2024

SUMMARY

INTRODUCTION

In the current context of industrial production, the efficient management of resources and the minimization of processing times are two of the main objectives for companies. The implementation of advanced scheduling techniques and Lean Manufacturing principles is crucial to achieve these goals and improve business competitiveness.

This paper focuses on the development of a scheduling algorithm in the MATLAB language, aimed at the optimal management of production orders. The code makes it possible to schedule the processing of materials on different types of machines: single, identical parallel and uniform parallel, based on the configuration of the production system.
The complexity of the production system is accentuated by the presence of various constraints, such as release dates and precedence constraints between activities, based on the processing cycle of each material.
It was decided to adopt the SPT (Shortest Processing Time) rule as the guiding principle for scheduling. The SPT rule, which prioritizes tasks with the shortest processing time, was chosen because of its ability to reduce the sum of total task completion times. If there are multiple tasks with equal processing times, the code is designed to start scheduling with the operation with the highest priority. This approach ensures that in addition to minimizing completion time, the sum of order penalties is also minimized.
The project illustrates the potential of the developed code, starting from the description of the Excel files used as input and continuing with practical examples that highlight the adaptability of the code to different configurations of the production system.
The output of the algorithm includes a summary table of the start and end times of processing of operations on the basis of which a Gantt chart is shown and a second table that reports for each order any hours of delay and the consequent penalty. Finally, the code shows the sum of the order completion time and the sum of the order penalties, providing an overall view of the performance of the production system.

This tool is particularly useful for businesses, as it allows them to benchmark between different configurations, helping them to identify the one that best suits their operating conditions.
Overall, the work presented makes a significant contribution to the efficient management of industrial production, with practical applications that can improve business production processes.

DESCRIPTION OF THE PROBLEM AND DEVELOPMENT OF THE ALGORITHM

Description of the scheduling problem

The scheduling problem addressed in this project can be formalized as follows:

$$1/\mathrm{P}/\mathrm{Q}\left|\ \mathrm{prec},\mathrm{r_j}\ \right|\sum \mathrm{C_j},\mathrm{P_j}$$

In this notation:

- $1/\mathrm{P}/\mathrm{Q}$: They indicate the presence of single, identical parallel, and uniform parallel machines.
- $\mathrm{prec},\mathrm{r_j}$: They indicate the presence of precedence constraints between a material's route operations and the release dates of an order, respectively.
- $\sum \mathrm{C_j},\mathrm{P_j}$: They indicate the objective functions, i.e. minimizing the sum of completion times ($\sum \mathrm{C_j}$) as the primary objective and sorting by order priorities ($\mathrm{P_j}$) as the secondary objective.

Single machines are independent production units that perform specific operations sequentially. Identical parallel machines are groups of machines that operate simultaneously with the same capacities and processing times. Finally, uniform parallel machines are machines that operate in parallel but with different processing speeds.
The production system has to comply with several constraints that make the scheduling problem more complex. Among these constraints are order release dates, which indicate the earliest time a task can start, and precedence constraints between tasks, which dictate the order in which operations should be executed.

The versatility of the code lies in the fact that it is able to optimize the scheduling of the tasks of any configuration that falls into the cases mentioned. So, you can decide to relax constraints (such as, for example, release dates) or modify the problem by inserting multiple copies of the same machine and allowing them to work at different speeds. These considerations can be seen in more detail in the input files that the code uses.

Description of Code Input Files

For the proper functioning of the scheduling algorithm developed in MATLAB, it is essential to provide a series of input data describing the production orders and the characteristics of the production system. This data is entered via structured Excel files, which contain all the information necessary for generating the optimal schedule.
The user of the executable must compile two input Excel files: the first is related to the production orders (called `INPUT_Orders.xlsx`), the second is related to the material processing cycle (called `INPUT_CoW.xlsx`).

Below, in the Table 1, a description of the columns in the first file is presented.

Table 1. Description of the features of the "INPUT_Orders.xlsx" file

Column	Description
ID_ORDER	A unique identifier for each production order.
ID_MATERIAL	Unique identifier of the material required for the order.
MATERIAL_QUANTITY	Quantity of material required to complete the order.
RELEASE_DATE	Release date, i.e. the earliest date from which the order can start processing.
DUE_DATE	Deadline date by which the order should be completed.
ORDER_PRIORITY	Order priority, which is used to determine the order of processing in the event of conflicts.

Below, the Table 2 with a description of the columns in the second file.

Table 2. Description of the features of the "INPUT_CoW.xlsx" file

Column	Description
ID_MATERIAL	Identifier of the material to be processed.
ID_OPERATION	Identifier of the specific operation to be performed on the material.
PREVIOUS_OPERATION	An identifier of the previous operation that was required before you could begin this operation.
ID_MACHINE	Identifier of the machine on which the operation is to be performed.
TIME_IN_HOURS	The time it takes to complete the operation, in hours.

However, in order to facilitate the reading of the data, the two files were unified, using a simple MATLAB code, cross-referencing the data using the foreign key *ID_MATERIAL*, common to both. The code used, named as `fileMergerCode.m` in Appendix 1. The file thus generated is named as `INPUT_OrdersAndCoW.xlsx`. The user of the code is to merge the files using it.

The code requires two input Excel files that are essential for its execution. Besides `INPUT_OrdersAndCoW.xlsx`, a second file is required, `INPUT_Configuration.xlsx`, which contains crucial data on the configuration of the production system. The latter file is critical in determining the type of production environment in use, which can range from single machines to parallel machines, both identical and uniform, or a combination of these. The nature of the production environment is deduced by analyzing the information provided in the file, the columns of which are described in detail in the Table 3.

Table 3. Description of the features of the "INPUT_Configuration.xlsx" file

Column	Description
ID_MACHINE_COPY	Identifier of the copy of a machine, belonging to a certain type of machine.
ID_MACHINE	Identifier of the machine on which the operation is to be performed.
MACHINE_PROCESSING_SPEED	The speed at which the machine processes operations.

In particular, it should be noted that while *ID_MACHINE* refers to the type of machine and the type of processing it can perform, *ID_MACHINE_COPY* refers to the unique identifier of each machine. So, if you see multiple copies referencing the same *ID_MACHINE*, then you have a problem with parallel machines. On the other hand, if only one copy appears, there is a problem with individual machines. The value of *MACHINE_PROCESSING_SPEED* allows the

problem to be distinguished into identical parallel machines (i.e. when all copies of a machine type have the same processing speed) or uniform parallel machines (i.e. when copies of a machine type have different processing speeds). In particular, the value of *MACHINE_PROCESSING_SPEED* indicates the speed at which a machine is able to perform an operation. This speed is expressed as a multiplier of the standard processing time, so if this value is equal to 1, the machine works at the standard processing time, if this is equal to 2 the machine works at twice the standard speed, halving the processing time, and so on.
For the purpose of code execution, *ID_MACHINE* is used as a foreign key, which creates relationships between the work cycles of the order materials and the production system in which they can be processed.

Description of the scheduling code

The developed code is a complex example of production order management using MATLAB. Its main purpose is to read data from Excel files, calculate the start and end times of operations, and visualize the production schedule in the form of a Gantt chart. The main points of the code are described below.

Uploading Input Files

The first step of the code is to upload the two Excel files that contain the production orders and the configuration of the machines:

```
% Load the "INPUT_OrdersAndCoW" file
file_path_orders = 'INPUT_OrdersAndCow.xlsx';
df_orders = readtable(file_path_orders);

% Load the "INPUT_Configuration" file
file_path_machines = 'INPUT_Configuration1.xlsx';
df_machines = readtable(file_path_machines, 'Sheet', 'Number_of_Machine');
```

Two files are read: one containing orders and operations (`INPUT_OrdersAndCow.xlsx`) and one with machine configurations (`INPUT_Configuration1.xlsx`), specifying the sheet `Number_of_Machine` for the second file.

Converting Dates

Order release and due dates are converted to *datetime* format for easy manipulation:

```
% Convert dates to datetime format
df_orders. RELEASE_DATE = datetime(df_orders. RELEASE_DATE, 'InputFormat',
 'yyyy-MM-dd');
df_orders. DUE_DATE = datetime(df_orders. DUE_DATE, 'InputFormat', 'yyyy-MM-
 dd');
```

Calculation of the start and end times of operations

The *calculate_schedule* function calculates the start and end times of each operation, considering the processing speed of the machines:

```
function df_orders = calculate_schedule(df_orders, df_machines)
```

This function:

- Initializes the start and end times for each operation.
- Use dictionaries to keep track of machine end times and orders.
- Calculates the total machining time for each operation.
- Sort operations with precedence constraints and the shortest machining time (SPT) rule.
- Determine the earliest time available for machines and assign operations to machines.

Sorting and Viewing Results

Orders are sorted by ID and by start time, and then the results are displayed:

```
% Sort the DataFrame by ID_ORDER and START_TIME
df_orders = sortrows(df_orders, {'ID_ORDER', 'START_TIME'});

% Display the results
disp(df_orders(:, {'ID_ORDER', 'ID_OPERATION', 'ID_MACHINE',
'ID_MACHINE_COPY', 'ID_MATERIAL', 'START_TIME', 'END_TIME'}));
```

Table of Contents and Penalties

The summary of operations is calculated and displayed, including any delay and penalties:

```
% Calculate and print the time when the last operation of each order
finishes
order_summary = varfun(@max, df_orders, 'InputVariables', {'DUE_DATE',
'END_TIME', 'ORDER_PRIORITY'}, 'GroupingVariables', 'ID_ORDER');

order_summary. LATE = order_summary.max_END_TIME >
order_summary.max_DUE_DATE;
order_summary. HOURS_LATE = hours(order_summary.max_END_TIME -
order_summary.max_DUE_DATE);
order_summary. HOURS_LATE(order_summary. HOURS_LATE < 0) = 0;
order_summary. PENALTY = order_summary. HOURS_LATE .*
order_summary.max_ORDER_PRIORITY;

% Rename the columns
order_summary. Properties.VariableNames{'max_DUE_DATE'} = 'DUE_DATE';
order_summary. Properties.VariableNames{'max_END_TIME'} = 'END_TIME';

% Display the summary table
disp(order_summary(:, {'ID_ORDER', 'DUE_DATE', 'END_TIME', 'LATE',
'HOURS_LATE', 'PENALTY'}));
```

Creating the Gantt Chart

The Gantt chart is created to graphically display the production schedule:

```
figure;
hold on;

orders = unique(df_orders.ID_ORDER);
colors = random_colors(length(orders));  % Generate a set of random colors
for the orders

% Initialize an array of handles for the legend
legend_handles = gobjects(length(orders), 1);
```

```
for i = 1:length(orders)
    order_id = orders{i};
    order_rows = df_orders(strcmp(df_orders.ID_ORDER, order_id), :);
    color = colors(i, :);  % Unique color for each order

    for j = 1:height(order_rows)
        machine_copy = order_rows.ID_MACHINE_COPY(j);
        start_time = datenum(order_rows. START_TIME(j));
        end_time = datenum(order_rows. END_TIME(j));

        y = find(strcmp(df_machines.ID_MACHINE_COPY, machine_copy));  %
Position of the machine copy on the y-axis

        % Draw a rectangle to represent the operation
        rectangle('Position', [start_time, y-0.4, end_time-start_time,
0.8], 'FaceColor', color, 'EdgeColor', 'k');

        % Add text to the bar
        center_time = start_time + (end_time - start_time) / 2;
        text(center_time, y, char(order_rows.ID_MATERIAL(j)),
'VerticalAlignment', 'middle', 'HorizontalAlignment', 'center', 'FontSize',
8, 'Color', 'white', 'BackgroundColor', 'black', 'Margin', 1);
    end

    % Create an invisible rectangle for the legend
    legend_handles(i) = plot(nan, nan, 's', 'MarkerEdgeColor', 'k',
'MarkerFaceColor', color);
end

% Configure the axes
set(gca, 'YTick', 1:length(df_machines.ID_MACHINE_COPY), 'YTickLabel',
cellstr(df_machines.ID_MACHINE_COPY), 'TickLabelInterpreter', 'none');
datetick('x', 'yyyy-mm-dd', 'keepticks');
xlabel('Date');
ylabel('Machine Copy');
title('Gantt Chart of Orders and Operations');
grid is;

% Add the legend
legend(legend_handles, orders, 'Location', 'bestoutside');

hold off;
```

This code:

- Generates a distinct color for each order.
- Draw rectangles that represent operations on the Gantt chart.
- Adds a legend and configures axes for clear display.

In summary, this MATLAB code, named as `solverCode.m` and reproduced in its entirety in

Appendix 2, reads production data, calculates processing times considering machine availability, visualizes results, and creates a Gantt chart to aid in the planning and management of production orders. It is a powerful tool for optimizing resource usage and reducing delays while providing a clear view of operations.

Description of Outputs

The developed MATLAB code generates several key outputs that provide a detailed overview of the production schedule and operations performance. The outputs are shown below.

1. **Table of Results of Operations**

The first table displayed by the code shows detailed information for each operation. In Table 4 there is a description of the columns displayed is shown.

Table 4. Description of the features of the Table of Results of Operations

Column	Description
ID_ORDER	Identifier of the order.
ID_OPERATION	Identifier of the operation within the order.
ID_MACHINE	Identifier of the machine assigned to the operation.
ID_MACHINE_COPY	Identifier of the specific copy of the machine used.
ID_MATERIAL	Identifier of the material processed.
START_TIME	Operation start time.
END_TIME	End time of the operation.

This table allows you to see exactly when and on which machine each operation is scheduled, providing a clear view of resource utilization.

2. **Order Summary Table**

The second table provides a summary of the performance of each order, with the following columns explained in Table 5.

Table 5. Description of the features of the Order Summary Table

Column	Description
ID_ORDER	Identifier of the order.
DUE_DATE	Expiration date of the order.
END_TIME	End time of the last operation of the order.
LATE	A Boolean indicator that indicates if the order is late from the due date.
HOURS_LATE	The number of hours of delay, calculated as the difference between the time the order ends and the due dates of the order itself.
PENALTY	Penalty calculated by multiplying the hours of delay by the priority of the order.

This summary provides an at-a-glance assessment of performance against lead times and potential penalties accrued due to delays.

3. **Sum of Order Completion Times**

The code calculates and prints the sum of order completion times. It is an important KPI of the overall efficiency of the production process.

4. **Total Penalty**

The sum of the order penalties is also calculated and printed. This value provides a measure of the financial impact of delays on production.

5. **Gantt Chart**

The visual output of the code is the Gantt chart, which graphically represents the production schedule. Each order is represented by a distinct color, and each operation is displayed as a rectangle positioned according to the start and end times. The axes of the diagram show time (x-axis) and machines (y-axis). The diagram also includes a legend that helps identify which order corresponds to each color.

This diagram is a powerful visual tool for understanding the sequence of operations, machine utilization, and identifying any overlaps or downtime.

Conclusions on Outputs

The outputs generated by the code provide a detailed and visual analysis of the production schedule. The combination of detailed tables and visual diagrams allows you to:

- Monitor and evaluate the performance of operations.
- Identify delays and their penalties.
- Optimize the use of machines.
- Improve order planning and management.

These outputs are critical for making informed decisions and improving the overall efficiency of the production process.

CODE APPLICATION EXAMPLES

After the description of the scheduling algorithm developed in MATLAB, the input files necessary for its operation and the outputs generated in response, we now move on to examine some practical cases. This chapter aims to demonstrate the effectiveness of the code through a concrete example, illustrating how data is used to generate an optimal schedule.

Configuration 1 (Standard)

Input Data

Initially, the two basic input files are presented: `INPUT_Orders.xlsx` (Table 6) and `INPUT_CoW.xlsx` (Table 7). 10 production orders and 7 distinct materials were assumed, each with its own processing cycle.

Table 6. INPUT_Orders.xlsx

ID_ORDER	ID_MATERIAL	MATERIAL_QUANTITY	RELEASE_DATE	DUE_DATE	ORDER_PRIORITY
OR01	MAT03	2	01/01/24 00:00	05/01/24 00:00	3
OR02	MAT02	8	05/01/24 00:00	09/01/24 00:00	1
OR03	MAT01	4	03/01/24 00:00	15/01/24 00:00	2
OR04	MAT02	7	01/01/24 00:00	05/01/24 00:00	2
OR05	MAT04	9	12/01/24 00:00	25/01/24 00:00	2
OR06	MAT01	10	02/01/24 00:00	10/01/24 00:00	4
OR07	MAT05	1	28/01/24 00:00	01/02/24 00:00	3
OR08	MAT04	6	18/01/24 00:00	02/02/24 00:00	2
OR09	MAT06	8	26/01/24 00:00	03/02/24 00:00	4
OR10	MAT07	12	03/01/24 00:00	04/02/24 00:00	3

Table 7. INPUT_CoW.xlsx

ID_MATERIAL	ID_OPERATION	PREVIOUS_OPERATION	ID_MACHINE	TIME_IN_HOURS
MAT01	OP1	None	MAC01	5
MAT01	OP2	OP1	MAC02	10
MAT01	OP3	OP2	MAC03	8
MAT02	OP1	None	MAC01	15
MAT02	OP2	OP1	MAC03	8
MAT03	OP1	None	MAC02	4
MAT03	OP2	OP1	MAC01	6
MAT03	OP3	OP2	MAC04	10
MAT04	OP1	None	MAC01	5
MAT04	OP2	OP1	MAC02	8
MAT04	OP3	OP2	MAC01	10
MAT04	OP4	OP3	MAC03	2
MAT05	OP1	None	MAC04	15

MAT05	OP2	OP1	MAC02	6
MAT05	OP3	OP2	MAC03	12
MAT06	OP1	None	MAC04	8
MAT06	OP2	OP1	MAC05	6
MAT07	OP1	None	MAC05	3
MAT07	OP2	OP1	MAC04	9
MAT07	OP3	OP2	MAC05	11

They are combined to form the unified file `INPUT_OrdersAndCoW.xlsx`, which is essential for the scheduling algorithm.

Subsequently, in Table 8, you can find the input file for the standard configuration of the production system, named as `INPUT_Configuration1.xlsx`.

Table 8. INPUT_Configuration1.xlsx

ID_MACHINE_COPY	ID_MACHINE	MACHINE_PROCESSING_SPEED
MAC01_COPY01	MAC01	1
MAC02_COPY01	MAC02	1
MAC03_COPY01	MAC03	1
MAC04_COPY01	MAC04	1
MAC05_COPY01	MAC05	1

This configuration is defined as standard because it is characterized by the presence of only one copy for each type of machine and all copies have a uniform processing speed of operations.

Scheduling results

Based on the above input data, the code returns the following results.

1. **Table of Results of Operations** (Table 9)

Table 9. Table of Results of Operation for Standard Configuration

ID_ORDER	ID_OPERATION	ID_MACHINE	ID_MACHINE_COPY	ID_MATERIAL	START_TIME	END_TIME
OR01	OP1	MAC02	MAC02_COPY01	MAT03	01-Jan-2024 00:00:00	01-Jan-2024 08:00:00
OR01	OP2	MAC01	MAC01_COPY01	MAT03	05-Jan-2024 09:00:00	05-Jan-2024 21:00:00
OR01	OP3	MAC04	MAC04_COPY01	MAT03	05-Jan-2024 21:00:00	06-Jan-2024 17:00:00
OR02	OP1	MAC01	MAC01_COPY01	MAT02	08-Jan-2024 19:00:00	13-Jan-2024 19:00:00
OR02	OP2	MAC03	MAC03_COPY01	MAT02	16-Jan-2024 19:00:00	19-Jan-2024 11:00:00
OR03	OP1	MAC01	MAC01_COPY01	MAT01	07-Jan-2024 23:00:00	08-Jan-2024 19:00:00
OR03	OP2	MAC02	MAC02_COPY01	MAT01	12-Jan-2024 03:00:00	13-Jan-2024 19:00:00
OR03	OP3	MAC03	MAC03_COPY01	MAT01	15-Jan-2024 11:00:00	16-Jan-2024 19:00:00
OR04	OP1	MAC01	MAC01_COPY01	MAT02	01-Jan-2024 00:00:00	05-Jan-2024 09:00:00
OR04	OP2	MAC03	MAC03_COPY01	MAT02	05-Jan-2024 09:00:00	07-Jan-2024 17:00:00

OR05	OP1	MAC01	MAC01_COPY01	MAT04	13-Jan-2024 19:00:00	15-Jan-2024 16:00:00
OR05	OP2	MAC02	MAC02_COPY01	MAT04	15-Jan-2024 16:00:00	18-Jan-2024 16:00:00
OR05	OP3	MAC01	MAC01_COPY01	MAT04	18-Jan-2024 16:00:00	22-Jan-2024 10:00:00
OR05	OP4	MAC03	MAC03_COPY01	MAT04	22-Jan-2024 10:00:00	23-Jan-2024 04:00:00
OR06	OP1	MAC01	MAC01_COPY01	MAT01	05-Jan-2024 21:00:00	07-Jan-2024 23:00:00
OR06	OP2	MAC02	MAC02_COPY01	MAT01	07-Jan-2024 23:00:00	12-Jan-2024 03:00:00
OR06	OP3	MAC03	MAC03_COPY01	MAT01	12-Jan-2024 03:00:00	15-Jan-2024 11:00:00
OR07	OP1	MAC04	MAC04_COPY01	MAT05	28-Jan-2024 16:00:00	29-Jan-2024 07:00:00
OR07	OP2	MAC02	MAC02_COPY01	MAT05	29-Jan-2024 07:00:00	29-Jan-2024 13:00:00
OR07	OP3	MAC03	MAC03_COPY01	MAT05	29-Jan-2024 13:00:00	30-Jan-2024 01:00:00
OR08	OP1	MAC01	MAC01_COPY01	MAT04	22-Jan-2024 10:00:00	23-Jan-2024 16:00:00
OR08	OP2	MAC02	MAC02_COPY01	MAT04	23-Jan-2024 16:00:00	25-Jan-2024 16:00:00
OR08	OP3	MAC01	MAC01_COPY01	MAT04	25-Jan-2024 16:00:00	28-Jan-2024 04:00:00
OR08	OP4	MAC03	MAC03_COPY01	MAT04	28-Jan-2024 04:00:00	28-Jan-2024 16:00:00
OR09	OP1	MAC04	MAC04_COPY01	MAT06	26-Jan-2024 00:00:00	28-Jan-2024 16:00:00
OR09	OP2	MAC05	MAC05_COPY01	MAT06	28-Jan-2024 16:00:00	30-Jan-2024 16:00:00
OR10	OP1	MAC05	MAC05_COPY01	MAT07	03-Jan-2024 00:00:00	04-Jan-2024 12:00:00
OR10	OP2	MAC04	MAC04_COPY01	MAT07	06-Jan-2024 17:00:00	11-Jan-2024 05:00:00
OR10	OP3	MAC05	MAC05_COPY01	MAT07	11-Jan-2024 05:00:00	16-Jan-2024 17:00:00

2. **Order Summary Table** (Table 10)

Table 10. Order Summary Table for Standard Configuration

ID_ORDER	DUE_DATE	END_TIME	LATE	HOURS_LATE	PENALTY
OR01	05-Jan-2024	06-Jan-2024 17:00:00	true	41	123
OR02	09-Jan-2024	19-Jan-2024 11:00:00	true	251	251
OR03	15-Jan-2024	16-Jan-2024 19:00:00	true	43	86
OR04	05-Jan-2024	07-Jan-2024 17:00:00	true	65	130
OR05	25-Jan-2024	23-Jan-2024 04:00:00	false	0	0
OR06	10-Jan-2024	15-Jan-2024 11:00:00	true	131	524
OR07	01-Feb-2024	30-Jan-2024 01:00:00	false	0	0
OR08	02-Feb-2024	28-Jan-2024 16:00:00	false	0	0
OR09	03-Feb-2024	30-Jan-2024 16:00:00	false	0	0
OR10	04-Feb-2024	16-Jan-2024 17:00:00	false	0	0

3. **Sum of Order Completion Times**

```
Sum of order completion times: 4449.00 hours
```

4. **Total Penalty**

```
Total penalty sum: 1114.00
```

5. **Gantt Chart** (Figure 1)

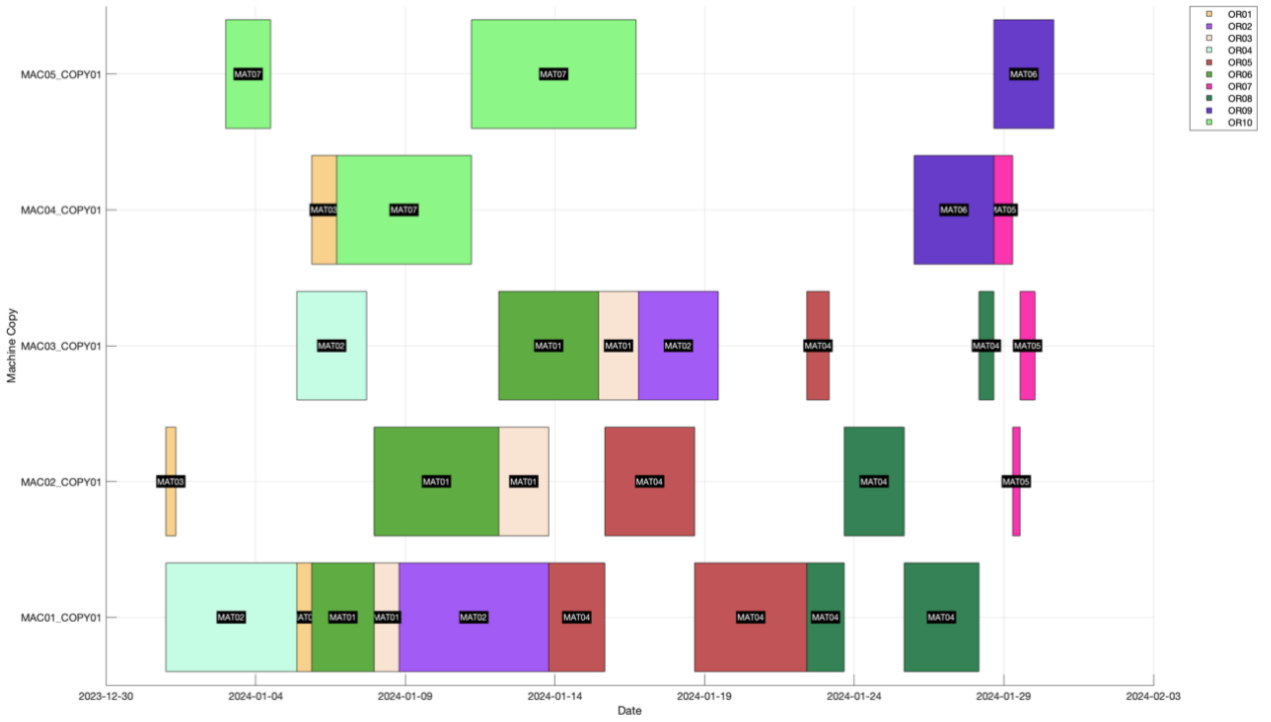

Figure 1. Standard Configuration Gantt chart

Analysis of the Impact of the Release Date Constraint on the Standard Configuration

In the previous section, the results obtained were generated by maintaining the constraint of the release date, which specifies the earliest date from which an order can start to be processed. In this section, this constraint is relaxed to evaluate the impact on the sum of the total order completion times. The hypothesis is that, by relaxing this constraint, the sum of completion times decreases, thus improving the overall efficiency of the production system.
The two input files used are identical to those described above, but with one significant change: all orders can start on January 1, 2024. This change allows us to observe how the order scheduling adapts in the absence of the release date constraint. For the execution of the code, please refer to the `INPUT_Orders_and_CoW_NOreleaseDate.xlsx` file, in which the release date of each order has been modified, and `INPUT_Configuration1.xlsx` which refers to the Standard Configuration previously mentioned.

Scheduling results

For simplicity's sake, here are only the most significant outputs generated by the code.

3. **Sum of Order Completion Times**

```
Sum of order completion times: 4228.00 hours
```

4. **Total Penalty**

```
Total penalty sum: 3256.00
```

Evaluating the Impact of the Release Date Constraint

Comparing the execution of the code in the two cases under analysis, i.e. with the same orders and configuration but with the presence or absence of the release date constraint, it can be seen that in the latter case the sum of the completion times decreased by 4.97% (from 4449 to 4228 hours). The sum of penalties has, however, increased; this explains why the first objective function applied by the code is the SPT scheduling rule, which aims to minimize the sum of completion times. The penalty increases because the code, in the scheduling of operations, takes into account the priority only as a second objective. Therefore, it intervenes in the code with the same processing time of several operations: in this case, quite remote, the one with the highest priority is assigned first.

Modifying Operational Configurations

The modification of the operational configurations of the production system is considered necessary to allow the system to effectively manage the scheduling of orders. From Figure 1 shows that machine 1 (*MAC01*) is very busy and consequently slows down all those processes that need it. For this reason, it was decided to test different operating configurations that allow the integration of identical and uniform parallel machines.
For each of the solutions that will follow, the input data relating to orders and work cycles are those reported in the Table 6. INPUT_Orders.xlsx. and in the Table 7. INPUT_CoW.xlsx. On the other hand, the operating configurations modified from the standard one will be presented on a case-by-case basis. For each of them, the last 4 outputs returned by the code are reported.

Based on the delays observed in the Table 10, the following changes have been made to the standard configuration.

Configuration 2

A copy of machine 1 is expected to be added. This configuration aims to distribute the workload across two identical parallel machines. Table 11 shows the input file for this new configuration of the production system, named as `INPUT_Configuration2.xlsx`. The outputs returned by the code are also shown below.

Table 11. INPUT_Configuration2.xlsx

ID_MACHINE_COPY	ID_MACHINE	MACHINE_PROCESSING_SPEED
MAC01_COPY01	MAC01	1
MAC01_COPY02	MAC01	1
MAC02_COPY01	MAC02	1
MAC03_COPY01	MAC03	1
MAC04_COPY01	MAC04	1
MAC05_COPY01	MAC05	1

2. **Order Summary Table** (Table 12)

Table 12. Order Summary Table for Configuration 2

ID_ORDER	DUE_DATE	END_TIME	LATE	HOURS_LATE	PENALTY
OR01	05-Jan-2024	02-Jan-2024 16:00:00	false	0	0
OR02	09-Jan-2024	15-Jan-2024 14:00:00	true	158	158
OR03	15-Jan-2024	12-Jan-2024 22:00:00	false	0	0
OR04	05-Jan-2024	07-Jan-2024 17:00:00	true	65	130
OR05	25-Jan-2024	21-Jan-2024 09:00:00	false	0	0
OR06	10-Jan-2024	11-Jan-2024 14:00:00	true	38	152
OR07	01-Feb-2024	30-Jan-2024 01:00:00	false	0	0
OR08	02-Feb-2024	24-Jan-2024 06:00:00	false	0	0
OR09	03-Feb-2024	30-Jan-2024 16:00:00	false	0	0
OR10	04-Feb-2024	14-Jan-2024 12:00:00	false	0	0

3. **Sum of Order Completion Times**

```
Sum of order completion times: 3871.00 hours
```

4. **Total Penalty**

```
Total penalty sum: 440.00
```

5. **Gantt Chart** (Figure 2)

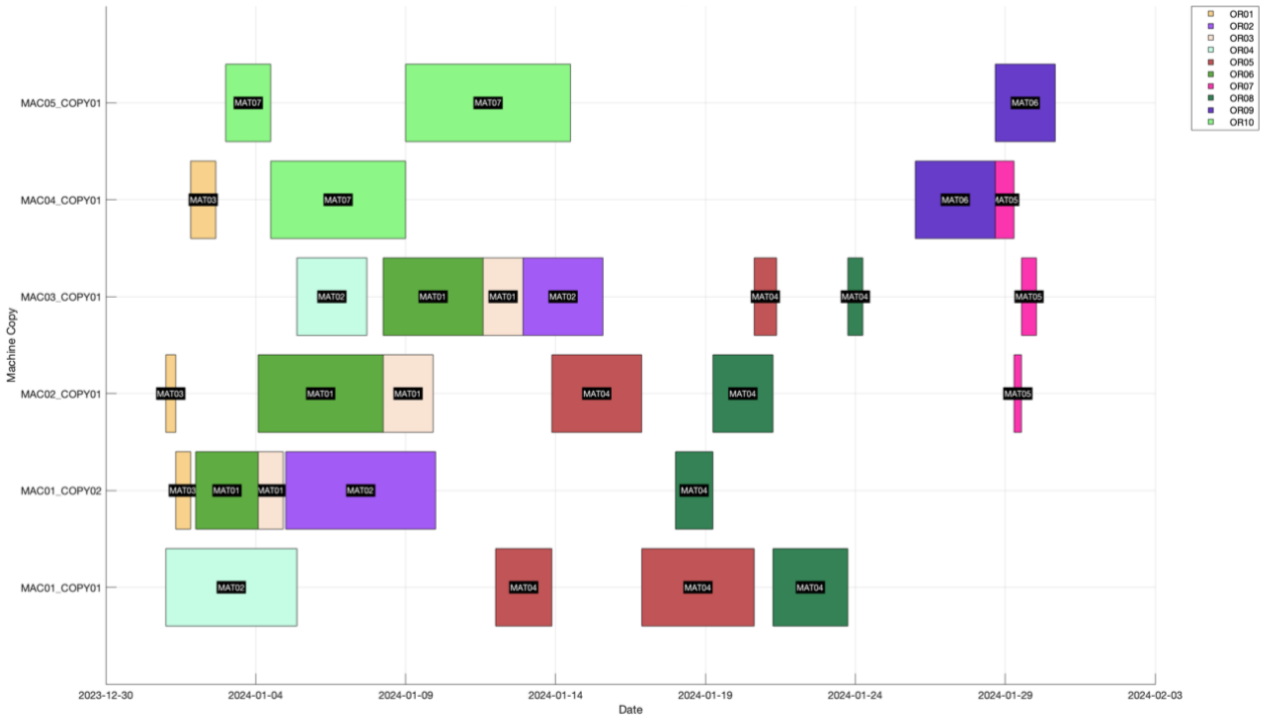

Figure 2. Configuration 2 Gantt chart

Configuration 3

By analyzing the outputs of Configuration 2, areas of intervention can be identified. It should be noted that the late orders are: *OR02* and *OR04* (having *MAT02* that is processed on *MAC01* and *MAC03*) and *OR06* (having *MAT01* that is processed on *MAC01*, *MAC02* and *MAC03*).
Therefore, it is planned to insert two copies of machine 1, one with unit speed and the other with double speed, and two copies of machine 3, both with double speed. Table 13 shows the input file for this new configuration of the production system, called `INPUT_Configuration3.xlsx`. The outputs returned by the code are also shown below.

Table 13. INPUT_Configuration3.xlsx

ID_MACHINE_COPY	ID_MACHINE	MACHINE_PROCESSING_SPEED
MAC01_COPY01	MAC01	2
MAC01_COPY02	MAC01	1
MAC02_COPY01	MAC02	1
MAC03_COPY01	MAC03	2
MAC03_COPY02	MAC03	2
MAC04_COPY01	MAC04	1
MAC05_COPY01	MAC05	1

2. **Order Summary Table** (Table 14)

Table 14. Order Summary Table for Configuration 3

ID_ORDER	DUE_DATE	END_TIME	LATE	HOURS_LATE	PENALTY
OR01	05-Jan-2024	02-Jan-2024 16:00:00	false	0	0
OR02	09-Jan-2024	08-Jan-2024 20:00:00	false	0	0
OR03	15-Jan-2024	10-Jan-2024 14:00:00	false	0	0
OR04	05-Jan-2024	04-Jan-2024 08:30:00	false	0	0
OR05	25-Jan-2024	18-Jan-2024 04:30:00	false	0	0
OR06	10-Jan-2024	09-Jan-2024 22:00:00	false	0	0
OR07	01-Feb-2024	29-Jan-2024 19:00:00	false	0	0
OR08	02-Feb-2024	22-Jan-2024 03:00:00	false	0	0
OR09	03-Feb-2024	30-Jan-2024 16:00:00	false	0	0
OR10	04-Feb-2024	14-Jan-2024 12:00:00	false	0	0

3. **Sum of Order Completion Times**

```
Sum of order completion times: 3399.00 hours
```

4. **Total Penalty**

```
Total penalty sum: 0.00
```

5. **Gantt Chart** (Figure 3)

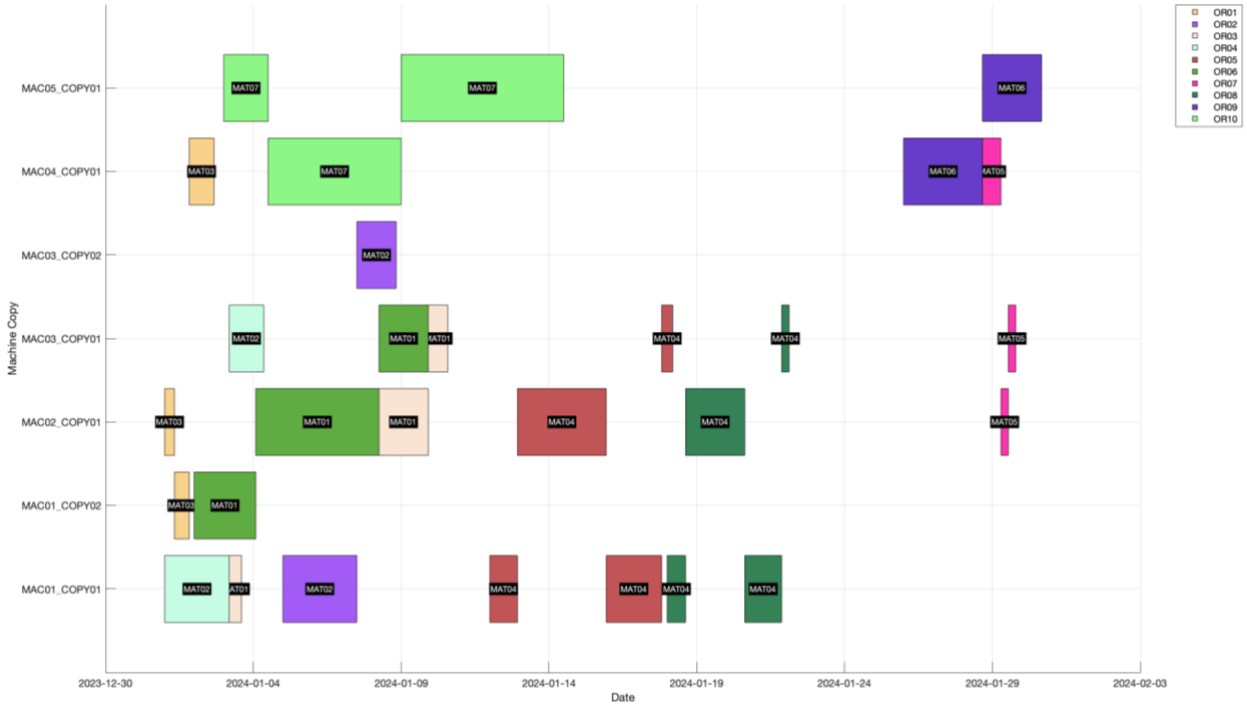

Figure 3. Gantt chart of Configuration 3

Analysis of the results

To evaluate the effectiveness of the different operating configurations, a benchmark was carried out between the tested configurations. The main KPIs used for comparison include:

- Sum of Order Completion Times
- Total Penalties

The benchmark results are summarized as follows: Figure 4. Comparison of Configurations.

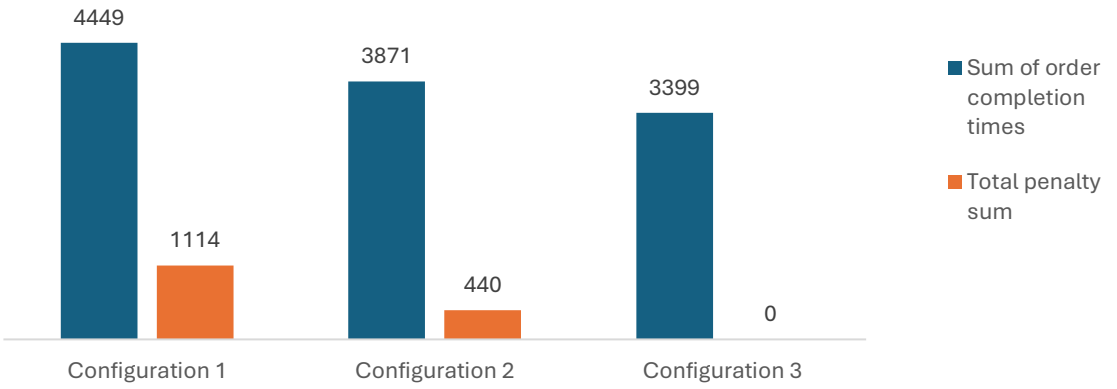

Figure 4. Comparison of Configurations

The comparison between the different operating configurations highlights the improvements achieved in terms of efficiency of the production system. The Figure 4 clearly shows how the changes made to the configurations have improved both the sum of order completion times and the total penalties.

- **Configuration 2** reduced the sum of order completion times by 13% compared to the standard configuration, from 4449 to 3871 hours. The total penalty has been reduced by 60% from 1114 to 440.
- **Configuration 3** achieved the best results, reducing the sum of order completion times by 24% compared to the standard configuration, from 4449 to 3399 hours. In addition, it completely eliminated penalties, highlighting how the inclusion of two copies of machine 1 (one with unit speed and the other with double speed) and two copies of machine 3 (both with double speed) can alleviate bottlenecks and improve the overall efficiency of this production system.

Recommendations for Optimal Configuration

Based on the results obtained, Configuration 3 may seem to be the optimal operating configuration for the analyzed production system. This configuration not only significantly reduces the sum of order completion times, but also completely eliminates penalties for delays, ensuring an overall improvement in system performance. Despite this, however, it would be necessary to analyze in detail the operating costs of the same and make a cost/benefit analysis to understand how economically convenient a solution of this type can actually be. This analysis demonstrates the importance of testing different operational configurations to identify the optimal solution, providing a methodological approach to improve the efficiency of production systems.

CONCLUSIONS

This paper described the development and implementation of a scheduling algorithm in MATLAB, designed to optimize production order management in complex industrial environments. The algorithm, which is based on the Shortest Processing Time (SPT) rule, has the ability to comply with complex operational constraints, such as release dates and precedence constraints, making it a valuable tool for companies looking to improve their competitiveness.

Analysis of the impact of the release date constraint showed that the code is robust to constraint relaxation.

The different operational configurations tested highlighted the importance of adapting the production system to specific operational needs. In particular, the addition of copies of machines and the variation of their processing speeds have led to significant improvements both in terms of reducing completion times and eliminating penalties. For example, Configuration 3 performed best, demonstrating the effectiveness of the approach in alleviating bottlenecks and improving overall efficiency.
Although the optimal operational configuration appears to be one with additional copies and variable machine speeds, it is essential to conduct a detailed cost-benefit analysis to assess the economic feasibility of such changes.

The generated Gantt charts provide a clear visual representation of the production schedule, making it easy to identify bottlenecks and areas for improvement. These visual tools are essential for effective planning and management of production orders, allowing managers to make informed decisions.

This paper demonstrated how the scheduling algorithm developed in MATLAB can be used to test different operational configurations and identify the optimal solution for specific production conditions.
In summary, the work presented offers a significant contribution to the efficient management of industrial production. The developed scheduling algorithm represents a powerful tool for improving production processes, allowing companies to identify and implement optimal operational configurations. Future research could further deepen the economic analysis of the different configurations.

APPENDIX

Appendix 1

```
% File paths
orders_file_path = 'INPUT_Orders.xlsx';
cow_file_path = 'INPUT_CoW.xlsx';

% Load Excel files into tables
orders_df = readtable(orders_file_path);
cow_df = readtable(cow_file_path);

% Merge tables based on the ID_MATERIAL column
merged_df = outerjoin(orders_df, cow_df, 'Keys', 'ID_MATERIAL', 'MergeKeys',
true);

% Save the merged table into a new Excel file
merged_file_path = '/Users/Downloads/INPUT_OrdersAndCow.xlsx';
writetable(merged_df, merged_file_path);

% Display the path of the merged file
disp(['Merged file saved at: ', merged_file_path]);
```

Appendix 2

```
CLC;
close all;
clear all;

% LOADING THE INPUT EXCEL FILES

% Load the "INPUT_OrdersAndCoW" file
file_path_orders = 'INPUT_OrdersAndCow.xlsx';
df_orders = readtable(file_path_orders);

% Load the "INPUT_Configuration" file
file_path_machines = 'INPUT_Configuration1.xlsx';
df_machines = readtable(file_path_machines, 'Sheet', 'Number_of_Machine');

% PROBLEM SOLVER

% Convert dates to datetime format
df_orders. RELEASE_DATE = datetime(df_orders. RELEASE_DATE, 'InputFormat',
'yyyy-MM-dd');
df_orders. DUE_DATE = datetime(df_orders. DUE_DATE, 'InputFormat', 'yyyy-MM-
dd');

% Function to calculate start and end times of operations, considering
MACHINE_PROCESSING_SPEED
function df_orders = calculate_schedule(df_orders, df_machines)
    df_orders. START_TIME = NaT(height(df_orders), 1);
    df_orders. END_TIME = NaT(height(df_orders), 1);
    df_orders.ID_MACHINE_COPY = strings(height(df_orders), 1);

    % Dictionary to keep track of the end times of machine copies
    machine_end_times = containers. Map(df_machines.ID_MACHINE_COPY,
repmat({datetime('0001-01-01', 'InputFormat', 'yyyy-MM-dd')},
height(df_machines), 1));
    % Dictionary to keep track of the end times of orders
    order_keys = unique(df_orders.ID_ORDER);
    order_end_times = containers. Map(order_keys, repmat({datetime('0001-01-01',
'InputFormat', 'yyyy-MM-dd')}, length(order_keys), 1));

    % Calculate the total processing time for each operation
    df_orders. PROCESSING_TIME = df_orders. TIME_IN_HOURS .* df_orders.
MATERIAL_QUANTITY;

    % Sort operations to respect precedence constraints and the SPT rule
    df_orders = sortrows(df_orders, {'RELEASE_DATE', 'PREVIOUS_OPERATION',
'PROCESSING_TIME', 'ORDER_PRIORITY'}, {'ascend', 'ascend', 'ascend',
'descend'});

    for i = 1:height(df_orders)
        release_date = df_orders. RELEASE_DATE(i);

        if ismissing(df_orders. PREVIOUS_OPERATION(i)) || strcmp(df_orders.
PREVIOUS_OPERATION(i), 'None')
            start_time = max(release_date,
order_end_times(char(df_orders.ID_ORDER(i))));
        else
            prev_end_time = df_orders. END_TIME(strcmp(df_orders.ID_ORDER,
df_orders.ID_ORDER(i)) & strcmp(df_orders.ID_OPERATION, df_orders.
PREVIOUS_OPERATION(i)));
```

```
            if isempty(prev_end_time)
                prev_end_time = release_date;
            end
            start_time = max(prev_end_time,
order_end_times(char(df_orders.ID_ORDER(i))));
        end

        % Find the first available machine copy
        available_machines = df_machines(strcmp(df_machines.ID_MACHINE,
df_orders.ID_MACHINE(i)), :);
        available_machine = '';
        for j = 1:height(available_machines)
            if machine_end_times(char(available_machines.ID_MACHINE_COPY(j))) <=
start_time
                available_machine = available_machines.ID_MACHINE_COPY(j);
                break;
            end
        end
        if isempty(available_machine)
            % If no machine is available, select the one that will be available
first
            min_end_time = datetime('9999-12-31', 'InputFormat', 'yyyy-MM-dd');
% Initialize with a very future date
            for j = 1:height(available_machines)
                machine_copy = char(available_machines.ID_MACHINE_COPY(j));
                end_time = machine_end_times(machine_copy);
                if end_time < min_end_time
                    min_end_time = end_time;
                    available_machine = machine_copy;
                end
            end
            start_time = max(start_time, min_end_time);
        end

        % Get the processing speed of the available machine copy
        processing_speed = df_machines.
MACHINE_PROCESSING_SPEED(strcmp(df_machines.ID_MACHINE_COPY,
available_machine));
        adjusted_operation_time = df_orders. PROCESSING_TIME(i) /
processing_speed;

        df_orders. START_TIME(i) = start_time;
        df_orders. END_TIME(i) = start_time + hours(adjusted_operation_time);
        df_orders.ID_MACHINE_COPY(i) = available_machine;

        % Update the end time of the machine copy
        machine_end_times(char(available_machine)) = df_orders. END_TIME(i);
        % Update the end time of the order
        order_end_times(char(df_orders.ID_ORDER(i))) = df_orders. END_TIME(i);
    end
end

% Calculate start and end times
df_orders = calculate_schedule(df_orders, df_machines);

% Sort the DataFrame by ID_ORDER and START_TIME
df_orders = sortrows(df_orders, {'ID_ORDER', 'START_TIME'});

% Display the results
disp(df_orders(:, {'ID_ORDER', 'ID_OPERATION', 'ID_MACHINE', 'ID_MACHINE_COPY',
'ID_MATERIAL', 'START_TIME', 'END_TIME'}));
```

```
% Calculate and print the time when the last operation of each order finishes
order_summary = varfun(@max, df_orders, 'InputVariables', {'DUE_DATE',
'END_TIME', 'ORDER_PRIORITY'}, 'GroupingVariables', 'ID_ORDER');

order_summary. LATE = order_summary.max_END_TIME > order_summary.max_DUE_DATE;
order_summary. HOURS_LATE = hours(order_summary.max_END_TIME -
order_summary.max_DUE_DATE);
order_summary. HOURS_LATE(order_summary. HOURS_LATE < 0) = 0;
order_summary. PENALTY = order_summary. HOURS_LATE .*
order_summary.max_ORDER_PRIORITY;

% Rename the columns
order_summary. Properties.VariableNames{'max_DUE_DATE'} = 'DUE_DATE';
order_summary. Properties.VariableNames{'max_END_TIME'} = 'END_TIME';

% Display the summary table
disp(order_summary(:, {'ID_ORDER', 'DUE_DATE', 'END_TIME', 'LATE', 'HOURS_LATE',
'PENALTY'}));

% Calculate and print the total completion time of ID_ORDERS
total_completion_time = sum(order_summary. END_TIME - df_orders.
RELEASE_DATE(1));
fprintf('Sum of order completion times: %.2f hours\n',
hours(total_completion_time));

% Calculate and print the total penalty sum
total_penalty = sum(order_summary. PENALTY);
fprintf('Total penalty sum: %.2f\n', total_penalty);

% Function to generate distinct colors and not in gradient
function colors = random_colors(n_colors)
    rng(4); % Set the seed for reproducibility
    colors = rand(n_colors, 3); % Generate random colors
    % Apply a modification to make them pastel
    colors = colors * 0.8 + 0.2;
end

% CREATING THE GANTT CHART

figure;
hold on;

orders = unique(df_orders.ID_ORDER);
colors = random_colors(length(orders));  % Generate a set of random colors for
the orders

% Initialize an array of handles for the legend
legend_handles = gobjects(length(orders), 1);

for i = 1:length(orders)
    order_id = orders{i};
    order_rows = df_orders(strcmp(df_orders.ID_ORDER, order_id), :);
    color = colors(i, :);  % Unique color for each order

    for j = 1:height(order_rows)
        machine_copy = order_rows.ID_MACHINE_COPY(j);
        start_time = datenum(order_rows. START_TIME(j));
        end_time = datenum(order_rows. END_TIME(j));

        y = find(strcmp(df_machines.ID_MACHINE_COPY, machine_copy));  % Position
of the machine copy on the y-axis
```

```
        % Draw a rectangle to represent the operation
        rectangle('Position', [start_time, y-0.4, end_time-start_time, 0.8],
'FaceColor', color, 'EdgeColor', 'k');

        % Add text to the bar
        center_time = start_time + (end_time - start_time) / 2;
        text(center_time, y, char(order_rows.ID_MATERIAL(j)),
'VerticalAlignment', 'middle', 'HorizontalAlignment', 'center', 'FontSize', 8,
'Color', 'white', 'BackgroundColor', 'black', 'Margin', 1);
    end

    % Create an invisible rectangle for the legend
    legend_handles(i) = plot(nan, nan, 's', 'MarkerEdgeColor', 'k',
'MarkerFaceColor', color);
end

% Configure the axes
set(gca, 'YTick', 1:length(df_machines.ID_MACHINE_COPY), 'YTickLabel',
cellstr(df_machines.ID_MACHINE_COPY), 'TickLabelInterpreter', 'none');
datetick('x', 'yyyy-mm-dd', 'keepticks');
xlabel('Date');
ylabel('Machine Copy');
title('Gantt Chart of Orders and Operations');
grid is;

% Add the legend
legend(legend_handles, orders, 'Location', 'bestoutside');

hold off;
```