mathematics

MDPI

Article

Automated and Optimized Scheduling for CNC Machines

Guilherme Sousa Silva Martins [1], M. Fernanda P. Costa [1] and Filipe Alves [2,*]

[1] Centre of Mathematics, University of Minho, 4710-057 Braga, Portugal; pg52214@alunos.uminho.pt (G.S.S.M.); mfc@math.uminho.pt (M.F.P.C.)
[2] DTx—Digital Transformation CoLAB, University of Minho, 4800-058 Guimarães, Portugal
* Correspondence: filipe.alves@dtx-colab.pt

Abstract

This work presents the design and implementation of an automated, digital, and modular system to address a real-world industrial challenge: the automation and optimization of production schedules for Computer Numerical Control (CNC) machines in a factory in Portugal. The goal is to replicate and enhance the existing manual scheduling process by integrating multiple data sources and formulating a general Mixed-Integer Linear Programming (MILP) model with constraints. This model can be solved using MILP optimization methods to produce efficient scheduling solutions that minimize machine downtime, reduce tool change frequency, and lower operator workload. The proposed system is implemented using open-source Python abstraction interfaces (Python-MIP), employing state-of-the-art of MILP optimization solvers such as CBC and HiGHS for solution validation. The system is designed to accommodate a wide range of constraints and operational factors, which can be switched on or off as needed, thereby enhancing its flexibility and decision-support capabilities. Additionally, a user-friendly graphical application is developed to facilitate the input of specific scheduling data and constraints, enabling flexible and efficient formulation of diverse scheduling scenarios. The proposed system is validated through multiple case studies, demonstrating its effectiveness in optimizing industrial CNC scheduling tasks and providing a scalable, practical tool for real-world factory operations.

Keywords: CNC machines; automation; smart-scheduling; optimization; software; MILP; solver; turn on–off functionality; factory

MSC: 90-04; 90-10; 90C05; 90C11

check for updates

Academic Editor: Andrea Scozzari

Received: 8 July 2025
Revised: 4 August 2025
Accepted: 12 August 2025
Published: 15 August 2025

Citation: Martins, G.S.S.; Costa, M.F.P.; Alves, F. Automated and Optimized Scheduling for CNC Machines. *Mathematics* **2025**, *13*, 2621. https://doi.org/10.3390/math13162621

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With technological advancements and the environmental challenges society faces today, the demand for increasingly ambitious and sustainable projects has become inevitable, particularly within the industrial sector. While these innovations may give us new tools to execute operations in less time and bring exciting opportunities, they also introduce new challenges, where wrong decisions can be damaging to companies, negatively impacting productivity and profitability [1].

One of the most relevant industrial problems we face is optimizing the production scheduling, in which multiple jobs must be assigned and processed on specialized machines while following the company constraints [2]. When the number of machines is substantively increased and with more complex workplace requirements, like managing employees

responsible for supervising large-scale operations, the use of rigorous mathematical techniques and the most efficient software programs in order to achieve the desirable goal is necessary. For this reason, the Operational Research methodology and its mathematical optimization algorithms become highly relevant to solve these real industrial problems [3].

One modern version of the scheduling problem is its application of industrial jobs and operations on CNC machines: "automated control machines that have a command function, a drive/motion mechanism, and a feedback system". CNC machines are typically used to "manufacture parts by shaping solid materials into desired forms" [4]. The combination of software development with the construction of mathematical programming models that correctly describe the real problem is what experts usually use.

Nowadays, many industrial companies and factories make use of CNC machinery to improve their production environments and increase efficiency. But, usually, those companies and factories do not take advantage of the recent technological and scientific advancements to improve their operations assignments on CNC machines. On the one hand, some of these companies have not yet implemented software applications that could help them optimize their production lines. On the other side, most of their employees do not have the necessary knowledge skills to define and to code the mathematical programming problems, which will require an expert in the area of mathematical optimization to set up and solve the models. This is the case of a factory in Portugal that has a set of available CNC machines and workers with different skills.

Thus, in this work, we propose to create a general MILP model capable of describing various CNC scheduling scenarios, and to study the development of a no-code and user-friendly software program capable of reading autonomously Excel files, with a functionality that enables switching on–off specific information and/or constraints and is able to generate schedule charts that minimize CNC machine downtime and lower operator workload; all of these without requiring the user to write a single line of code.

This paper is organized as follows. Section 2 provides a review of the relevant literature. Section 3 presents the methodology and the mathematical formulation of the Scheduling Problem. Section 4 describes the application and software development. Section 5 presents the results and discussion. Section 6 concludes the paper and outlines directions for future work. Furthermore, Appendices A.1 and A.2 contain the notation and the screenshots from the developed system, respectively.

2. Literature Review

A scheduling problem involves the assignment of tasks to the available resources (machines and workers) over time while satisfying specific constraints and optimizing certain objectives [5]. Scheduling problems are called *deterministic* when all the input data is known with certainty and does not change over time. When production environments are subject to sources of uncertainty or randomness, the scheduling problem is classified as a *stochastic* one [5].

According to [5], a deterministic scheduling problem can be described by a triplet $\alpha|\beta|\gamma$, where the α field indicates the machine environment, the β field shows details of processing characteristics and its specific constraints, and lastly, the γ field describes the objective function to be minimized. The combination of those three fields creates different types of scheduling problems, like the Flexible Job Shop variant. Some of the most relevant machine environments that can be specified in the α field are:

- **Single machine**: This is the simplest form of machine environment in which there is only a single machine available, each job involves just one operation, and the processing time for each job is deterministic [6];

- **Parallel machine**: This follows the rules of the single machine version, but here there are m machines, where each job has to be processed mandatorily on each one of the m machines;
- **Job Shop Scheduling Problem (JSSP)**: This is like the parallel machine version, but with the addition that each job has its own predetermined processing order to follow. There are different versions of job shops in which each job may visit each machine at most once or in cases where it may visit each machine more than once.
- **Flow shop**: This is similar to the JSSP, but all jobs have to follow the same predefined processing order.
- **Open shop**: This is an extension of the JSSP, where some of the processing times may be zero, and there are no restrictions with regard to the processing order of each job through the machine environment (it is the responsibility of the factory to decide the previous factors).
- **Flexible**: This is an extension to the previous machine environments, where there are machine alternatives for the same job, instead of a fixed machine.

The constraints specified in the β field may include multiple entries, such as release dates and precedences between jobs.

When different types of machine environments are combined with different kinds of constraints and new ones are added to the scheduling, for example, workers to supervise during machine execution, the problem generally becomes *NP-hard*. This means no polynomial-time algorithm is known that solves the problem optimally, and metaheuristics or relaxation methods are often used to find feasible solutions [7].

Scheduling problems can be described through MILP formulations, where their optimal solutions can be found efficiently through optimization solvers [3,5,8]. Relevant examples are modern MILP solvers, such as Gurobi or CPLEX, and **open-source** code such as COIN-OR CBC, Python-MIP, and HiGHS, which are equipped with high-performance algorithms to solve large-scale MILP problems [9–13].

The *Branch-and-Cut* algorithm is currently the most popular used architecture for a MILP solver [8], along with efficient techniques, like the *Lagrangian relaxation*, *Benders*, and *Dantzig–Wolfe decompositions*; *Genetic Algorithms*; metaheuristics, such as the *Large Neighborhood Search (LNS)* and *Simulated Annealing*; and cut generators [8]. Moreover, with the advancements from machine learning [8] and quantum computing [14], solvers can take advantage of their unique characteristics to improve optimization algorithms.

3. Methodology and Scheduling Problem

This work adopted a quantitative and computational approach, focusing on formulating rigorously the scheduling problem as a mathematical model and developing a software web application that could optimize the desired schedules autonomously, without requiring programming skills from the user. Therefore, the applied methodology consists of the following stages:

- Formulation of the mathematical model;
- Software design and implementation;
- Computational testing and results analysis.

To simplify the exposition of the deterministic scheduling problem, we first introduce all relevant data and decision variables. Next, we present the objective function along with all associated constraints.

3.1. Minimal Information

Although a wide range of information can be used to define a CNC scheduling problem, there is a *minimal* set of input data that must be known in advance to describe the jobs and machines of the problem [5–7,11]:

- Set of **jobs**: $j \in \mathcal{J} = \{1, \ldots, n\}$;
- Set of **machines**: $i \in \mathcal{M} = \{1, \ldots, m\}$;
- The non-negative amount of **processing time** (p_{ij}) required by job j in machine i.

3.2. Optional Scheduling Information

After the minimal data is defined, it is possible to add extra information related to its jobs and schedules, add workers, priorities, among others. Consequently, its complexity and the necessary time to find an optimal solution increase. The list below shows a set of extra attributes/data that can be added to the formulation, as follows:

- (Optional) The **initial time**, t_0, for the scheduling problem;
- (Optional) The **time horizon** (T_{max});
- Set of **workers**: $k \in \mathcal{W} = \{1, \ldots, w\}$;
- If worker k has the necessary **skills** to operate machine i (binary value): $skill_{ki} \in \{0, 1\}$;
- The **release date** ($r_j \in T$) at which job j is available for processing;
- The time at which the processing of job j is due to be completed: **due date** ($d_j \in T$), also known as *deadline*;
- **Job precedences**: if all operations of job j must be completed before any operation of job j' can start (binary value): $U_{jj'} \in \{0, 1\}$;
- The **processing order** of job j (without repetition), $\mathcal{O}_j = (o^j_1, o^j_2, \ldots, o^j_{|m|})$, which indicates the machine that will receive the r-operation of job j;
- **Working interval schedules** for worker k: $WS_k = \{(ws^k_{start}, ws^k_{end})_1, (ws^k_{start}, ws^k_{end})_2 \ldots\}$, organized in chronological order;
- **Maintenance schedules** for machine i: $MS_i = \{(ms^i_{start}, ms^i_{end}), \ldots\}$, organized in chronological order;
- **Priority** (P_j): weight/importance of completion for job j.

Usually, time is considered continuous; however, for this software implementation, time is **discrete**, modeled as a sequence of equally separated time steps, where $t \in T = \{t_0, t_0 + 1, \ldots, T_{max}\}$, which must be transformed into $T = \{0, 1, 2, \ldots, T_{max} - t_0\}$. All remaining attributes also need to be transformed so that all data is correct for the same time range as the new T set.

When t_0 is not given to the program, it will be defined by default as 8 AM (usual start time for a work day); while when T_{max} is not given by the user to the application, then the program will calculate it as the usual 8 daily work hours.

3.3. Decision Variables

When time is divided into **discrete** values, we have 3 types of *decision variables* or *parameters* :

- $x_{ijt} \in \{0, 1\}$ (binary variable): Shows if machine i starts to process job j at instant t;
- $y_{ijkt} \in \{0, 1\}$ (binary variable): Shows if worker k starts to operate task j on machine i at time t (Optional when the program does not have any information about its workers);
- C_j: The total makespan for all operations from job j.

The maximum number of decision variables in this case can be variable, depending on the initial time, t_0, and its time horizon, T_{max}. A bad discretization of time can result in an absurd number of unnecessary binary variables to solve the problem.

3.4. Objective Function

The objective function must be able to minimize the time when the last task finishes [2,11,15]. Therefore, if the priorities P_j of each job $j \in \mathcal{J}$ are given to the user/program (warning that the priorities are *optional data*), the objective function must be able to minimize the sum of the product between the makespans of each job j (C_j) and their respective priorities:

$$\text{minimize } z = \sum_{j \in \mathcal{J}} P_j \times C_j \tag{1}$$

However, if priorities P_j are not given (in other words, $P_j = 1, \forall j \in \mathcal{J}$), then the objective function becomes the simple sum of *makespans* from all jobs j:

$$\text{minimize } z = \sum_{j \in \mathcal{J}} C_j \tag{2}$$

3.5. Mandatory Constraints

The mandatory constraints are those defined using only the minimal data required from the scheduling problem to ensure it remains solvable [2,11,15]:

- Each task/operation of job j must be executed **only once** and start each time on a machine at an **exact** time (if job j has a processing time):

$$\sum_{t \in T} x_{ijt} = 1, \quad \forall j \in \mathcal{J}, i \in \mathcal{M}, p_{ij} \neq 0 \tag{3}$$

$$\sum_{t \in T} x_{ijt} = 0, \quad \forall j \in \mathcal{J}, i \in \mathcal{M}, p_{ij} = 0 \tag{4}$$

  From now on, whenever any constraint uses information related to schedules, their time values must be corrected according to the initial time, t_0, in order to make the search for the optimal solution start at time $t = 0$. Moreover, the following constraints can only work with processes with a defined processing time ($p_{ij} > 0$).
- A machine can only receive one job at a time and run it until its processing time is completed (which also means that two jobs j and j' cannot be executed at the same time in the same machine i). So, a machine can only receive a new job after the first one has been concluded. This can be verified for each discrete time t through the auxiliary variable ao_{ijt} which can tell if operation j is active on machine i at the specific time t:

$$ao_{ijt} = \sum_{t_s = \max\{0, t - p_{ij} + 1\}}^{t} x_{ijt_s}, \quad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \in T$$

  The variable ao_{ijt} will inform if operation j on machine i is active at time t by verifying the values of x_{ijt} for the $p_{ij} - 1$ previous discrete times of t, inclusive. As it will make use of values from the variable x_{ijt}, ao_{ijt} will just verify if job j has ever started on machine i during the previous interval of time. One important aspect of this variable is that when a task from job j on machine i finishes at a specific discrete time t, the value of ao_{ijt} must be equal to 0, since the task becomes inactive, leaving that time to a (valid) job that has not started its processing on machine i.
  In order to avoid two jobs overlapping each other in the same machine (which also means that two jobs on the same machine cannot be active at the same time), the following constraint ensures that no more than one job can be active in the same machine at instant t:

$$\sum_{j \in \mathcal{J}} ao_{ijt} \leq 1, \quad \forall i \in \mathcal{M}, t \in T$$

Thus, this constraint can be expressed through the simple substitution of expression that defines ao_{ijt} as follow:

$$\sum_{j \in \mathcal{J}} \sum_{t_s = \max\{0, t - p_{ij} + 1\}}^{t} x_{ijt_s} \leq 1, \quad \forall i \in \mathcal{M}, t \in T \tag{5}$$

- The *makespan* of job j (C_j) must be greater or equal to the execution times of all operations of job j:

$$\sum_{i \in \mathcal{M}} \left(\sum_{t \in T} (t \times x_{ijt}) + p_{ij} \right) \leq C_j, \quad \forall j \in \mathcal{J} \tag{6}$$

3.6. Constraints from Optional Data

Unlike the mandatory constraints, the following constraints cannot be added to the problem formulation if some of the extra attributes are not provided in our database or the user wishes to *switch on–off* some of the available information [2,11,15].

First, the extra constraints related to ***jobs*** from Constraints (7) to (10) are introduced.

- No job can start until its previous one is completed. If defined in $U_{jj'}$ as equal to 1, then job j' can only start after job j has finished all of its defined operations:

$$\sum_{t \in T} [(t + p_{ij}) \times x_{ijt}] \leq \sum_{t \in T} (t \times x_{i'j't}), \quad \forall i, i' \in \mathcal{M}; j, j' \in \mathcal{J}; j \neq j'; U_{jj'} = 1 \tag{7}$$

- A job j can only start after its release time r_j:

$$\sum_{i \in \mathcal{M}} \sum_{t \in T,\, t < r_j} x_{ijt} = 0, \quad \forall j \in \mathcal{J} \tag{8}$$

- The completion time of a job j must not surpass its due date (d_j), which means its operations cannot start at a discrete time t that will end at a time after its due date:

$$\sum_{t \in T,\, t > \max\{0, d_j - p_{ij}\}} x_{ijt} = 0, \quad \forall i \in \mathcal{M}, j \in \mathcal{J} \tag{9}$$

- All operations/tasks from job j must be executed in the order of the sequence given by $\mathcal{O}_j$:

$$\sum_{t \in T} [(t + p_{ij}) \times x_{o^j_{r-1} jt}] \leq \sum_{t \in T} t \times x_{o^j_r jt}, \quad \forall j \in \mathcal{J}, o^j_r \in \mathcal{O}_j, r \in \{2, \ldots, m\} \tag{10}$$

Additional information about the defined maintenance schedules of its **machines** can impose the following constraint:

- While in maintenance time of machine i (MS_i), it cannot receive any jobs (until its maintenance has finished):

$$\sum_{t = \max\{t_0, ms^i_{start} - p_{ij} + 1\}}^{ms^i_{end} - 1} x_{ijt} = 0, \quad \forall i \in \mathcal{M}, j \in \mathcal{J}, (ms^i_{start}, ms^i_{end}) \in MS_i \tag{11}$$

It is assumed that no machine has a setup time when switching jobs (as we are working with hours only)! In other words, when a machine finishes one job, it can immediately start another without any delay. This assumption is reasonable in this context since the

scheduling is performed in units of whole hours, making short setup or transition times negligible or irrelevant for our scheduling model.

The last type of constraints from optional data is the one related to its **workers**. When a user gives data about its workers, it adds new decision variables, y_{ikt}, and the constraints (12) and (13). If a worker is added, but their skills and schedules are unknown, we admit that the worker has the ability to operate any machine at any time and the respective workers work full-time:

- Each machine operating a job must have at least one worker allocated. From the following formulation, it is the same as stating that the number of allocated workers to operation j on machine i must be equal to the number of times this exact operation has started:
$$\sum_{k \in \mathcal{W}} y_{ijkt} = x_{ijt}, \quad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \in T \tag{12}$$
- When a worker k is allocated to a process from job j on machine i, it must be there until the task finishes, being unable to work at two or more machines (or processes from the same machine) at the same time. From the other side, a worker can only start a task on a specific machine after the worker who was already supervising a task on the same machine finishes it.
Therefore, to help with the definition of this constraint, it was necessary to create auxiliary variables, aw_{kit}, that could tell if worker k is still working on machine i at discrete time t:
$$aw_{kit} = \sum_{j \in \mathcal{J}} \sum_{t_s = \max\{0, t - p_{ij} + 1\}}^{t} y_{ijkt_s}, \quad \forall k \in \mathcal{W}, i \in \mathcal{M}, t \in T$$
The previous variable verifies if the specific worker k has started any operations on machine i that could be active at instant t by analyzing if at the previous $p_{ij} - 1$ discrete time and at t, inclusive, any task has ever started for each job that must be executed on machine i.
Through this variable, we can build a constraint that guarantees the same worker cannot work two operations at the same time (either from the same machine, or from two different machines!) by counting how many tasks worker k is actively supervising:
$$\sum_{i \in \mathcal{M}} aw_{kit} \leq 1, \quad \forall k \in \mathcal{W}, t \in T$$
The final and simpler version of this constraint with just the decision variable y_{ijkt} can be built by using the expression that defines the variable aw_{kit}:
$$\sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{J}} \sum_{t_s = \max\{0, t - p_{ij} + 1\}}^{t} y_{ijkt} \leq 1, \quad \forall k \in \mathcal{W}, t \in T \tag{13}$$

Constraints (28) and (15) are required for the mathematical formulation if the program has access to the worker schedules and skills, respectively:

- Each worker k, when out of their working schedules WS_k, cannot be assigned to any available machine. To guarantee that no worker starts a task that might finish at a time out of their working hours, the following formulation had to consider the processing times from each available pair (job, machine):
$$\sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{J}} \left(\sum_{t \in T \setminus A_{kij}} y_{ijkt} \right) = 0, \quad \forall k \in \mathcal{W} \tag{14}$$

where $A_{kij} = \bigcup_{(ws^k_{start}, ws^k_{end}) \in WS_k} [ws^k_{start}, ws^k_{end} - p_{ij}]$

- A worker k can only operate a machine i if the worker has the necessary skills to manipulate it. If worker k does not have the skill to supervise machine i (i.e., $skill_{ki} = 0$), that worker cannot be assigned to any operation on that machine:

$$\sum_{j \in \mathcal{J}} y_{ijkt} \leq skill_{ki}, \quad \forall i \in \mathcal{M}, k \in \mathcal{W}, t \in T \tag{15}$$

It is assumed that the company manager is able to ensure that all tasks have the necessary raw materials to start their executions and that another person or company is responsible for the management of workers skilled to maintain the CNC machines. Consequently, the workers of the company are exclusively tasked with operating the machines and are not involved in maintenance or repair duties.

3.7. General MILP Model

For completeness, the full general mathematical formulation of the MILP model is defined as follows:

$$\underset{x_{ijt}, y_{ijkt}, C_j}{\text{minimize}}\, z = \sum_{j \in \mathcal{J}} P_j \times C_j \tag{16}$$

subject to:

$$\sum_{t \in T} x_{ijt} = 1, \quad \forall j \in \mathcal{J}, i \in \mathcal{M}, p_{ij} \neq 0 \tag{17}$$

$$\sum_{t \in T} x_{ijt} = 0, \quad \forall j \in \mathcal{J}, i \in \mathcal{M}, p_{ij} = 0 \tag{18}$$

$$\sum_{j \in \mathcal{J}} \sum_{t_s = \max\{0, t - p_{ij} + 1\}}^{t} x_{ijt_s} \leq 1, \quad \forall i \in \mathcal{M}, t \in T \tag{19}$$

$$\sum_{i \in \mathcal{M}} \left(\sum_{t \in T} (t \times x_{ijt}) + p_{ij} \right) \leq C_j, \quad \forall j \in \mathcal{J} \tag{20}$$

$$\sum_{t \in T} [(t + p_{ij}) \times x_{ijt}] \leq \sum_{t \in T} (t \times x_{i'j't}), \quad \forall i, i' \in \mathcal{M}; j, j' \in \mathcal{J}; j \neq j'; U_{jj'} = 1 \tag{21}$$

$$\sum_{i \in \mathcal{M}} \sum_{t \in T,\, t < r_j} x_{ijt} = 0, \quad \forall j \in \mathcal{J} \tag{22}$$

$$\sum_{t \in T,\, t > \max\{0, d_j - p_{ij}\}} x_{ijt} = 0, \quad \forall i \in \mathcal{M}, j \in \mathcal{J} \tag{23}$$

$$\sum_{t \in T} [(t + p_{ij}) \times x_{o^j_{r-1} jt}] \leq \sum_{t \in T} t \times x_{o^j_r jt}, \quad \forall j \in \mathcal{J}, o^j_r \in \mathcal{O}_j, r \in \{2, ..., m\} \tag{24}$$

$$\sum_{t = \max\{t_0, ms^i_{start} - p_{ij} + 1\}}^{ms^i_{end} - 1} x_{ijt} = 0, \quad \forall i \in \mathcal{M}, j \in \mathcal{J}, (ms^i_{start}, ms^i_{end}) \in MS_i \tag{25}$$

$$\sum_{k \in \mathcal{W}} y_{ijkt} = x_{ijt}, \quad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \in T \tag{26}$$

$$\sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{J}} \sum_{t_s = \max\{0, t - p_{ij} + 1\}}^{t} y_{ijkt} \leq 1, \quad \forall k \in \mathcal{W}, t \in T \tag{27}$$

$$\sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{J}} \left(\sum_{t \in T \setminus A_{kij}} y_{ijkt} \right) = 0, \quad \forall k \in \mathcal{W} \tag{28}$$

where $A_{kij} = \bigcup_{(ws^k_{start}, ws^k_{end}) \in WS_k} [ws^k_{start}, ws^k_{end} - p_{ij}]$

$$\sum_{j \in \mathcal{J}} y_{ijkt} \leq skill_{ki}, \quad \forall i \in \mathcal{M}, k \in \mathcal{W}, t \in T \tag{29}$$

$$x_{ijt} \in \{0,1\}, \quad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \in T$$

$$y_{ijkt} \in \{0,1\}, \quad \forall i \in \mathcal{M}, j \in \mathcal{J}, k \in \mathcal{W}, t \in T$$

$$C_j \geq 0, \quad \forall j \in \mathcal{J}$$

4. Application and Software Development

(**WARNING**: Due to data privacy, some private information and small parts of the code cannot be shared here!)

Since this work targets industrial enterprises whose employees and managers typically do not have experience with MILP formulation interfaces or advanced database methodologies, it was essential to develop a user-friendly system. Specifically, the system was designed to make it straightforward for users to upload data, select the relevant scheduling information, and generate the desired schedules without requiring specialized technical knowledge. Appendix A.2 shows the screenshots of the developed system.

The solution involved the development of a web application that could run in the *Microsoft Windows* operating system with an intuitive and user-friendly Graphical User Interface (GUI). To achieve this goal, the following software and languages were used:

- *Microsoft Excel*: Software for manual data sheet insertion, which does not require any knowledge or experience with relational data storage;
- *Python* (version: 3.10.15): Utilized for the adapted Excel compiler and the back-end programming, where the MILP formulation is created for the given data from the Excel file;
- *Flask* (version: 3.1.0): Enables the connection between back-end and front-end parts made in HTML and Python;
- *Python-MIP* (version: 1.15.0): The chosen MILP formulation constructor interface;
- *CBC* (version: October 28 2021) and *HiGHS* (version: 1.8.0): The available MILP solvers to solve the MILP problem [12]. Both solver versions are the ones available on the Python-MIP interface by default. Besides that, both solvers use the *Branch-and-Cut* algorithm (which can also be called *branch-and-cut-and-bound* [9]) to find the values that minimize the given MILP problem [8,16].

We have to clarify that both solvers use exact optimization methods, instead of heuristic or rule-based scheduling techniques to find a solution.

After the information related to the CNC machines and their available workers is filled out on the Excel file through specific instructions, the user can start the application. Its pages are responsive depending on the action that the user or the program itself takes. This complexity of actions between pages is systematized on the activity diagram presented in Figure 1.

Once the user initializes the application, the first page that appears on the screen is the *Startup/Upload page*. There, the user can upload an Excel file (*.xlsx*) and then proceed to the next step. If the program verifies the file does not have the extension *.xlsx*, it will not proceed to the "Option Selection" page, but it will refresh the actual one and warn the user to upload a valid format file.

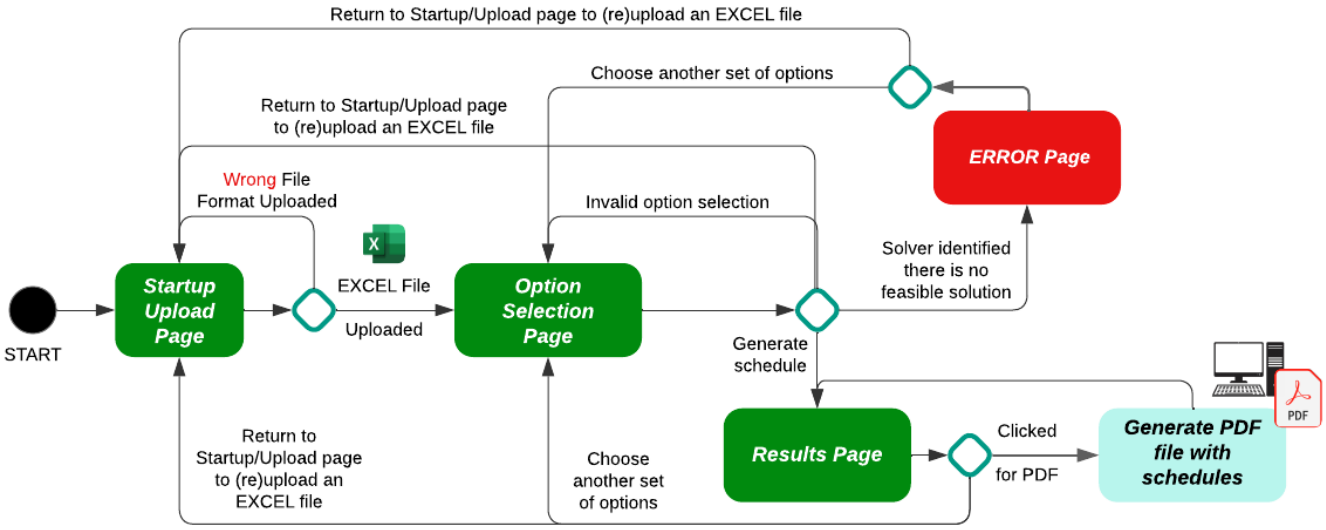


Figure 1. Activity diagram for interface interaction between pages.

When the uploaded file is validated by the application, the *Option Selection page* appears on the screen with an innovative functionality to *turn on–off* options related to the information provided by the uploaded file. This functionality enables the user to select:

- The machines, jobs, and workers (if available) that must be included in the machine (and worker) scheduling;
- The *optional/extra* information that can be added or deleted from the scheduling problem (release and due dates, priorities, defined schedules for workers, skills . . .)
- Which MILP solver must be used: CBC or HiGHS.

This page will only show the options that are related to the input data from the uploaded Excel file. For example, if the previous file does not contain any information related to employees, then the program will not allow the possibility to select workers for assignments, and if the user wishes to turn on or off the constraint related to them, like their skills and working hours.

When the selected information by the user is not enough for generating a schedule (like selecting no machines or no jobs), the application refreshes the page and warns the user to select an adequate minimum number of machines and jobs.

After selecting the necessary data, if the selected solver finds out the problem has an optimal feasible solution, the proposed system generates a page where the optimal scheduling for the CNC machine is shown through Gantt charts. When workers are introduced by the user and they are enough to produce a schedule for the enterprise employees, the solver will also return the results for the worker assignments, which are also shown on Gantt charts. In addition, this page provides functionality for users to conveniently export and download the generated charts in a PDF format. This feature enables easy sharing, printing, or further review of the scheduling results outside the web application of the proposed system.

5. Results and Discussion

In this section, a realistic example will be explored, inspired by a real-life case from a Portuguese industrial company. The main goal of this case is to improve the production scheduling for the primary parts of aeronautical structures, with a set of nine available MAKINO machines (with similar characteristics to the studied CNC machines, but adapted for the aeronautical materials and structures). The available MAKINO machines are

introduced on Table 1, along with their scheduled maintenance hours, except for machines A4 and A20.H.

Table 1. MAKINO CNC Machine information (and their maintenance hours).

Machine	Quantity	Created IDs	Maintenance Schedule
MAKINO A4	1	{MA4}	—
MAKINO A8.H	3	{MA8-1, MA8-2, MA8-3}	8:00–10:00
MAKINO A12	3	{MA12-1, MA12-2, MA12-3}	14:00–15:00
MAKINO A20.H	2	{MA20-1, MA20-2}	—

The expected jobs to be completed are denoted in the model as J1, J2, . . . , as shown in Tables 2 and 3.

Table 2. Job data: release/due hours and processing details.

Job	Release Hour	Due Time	Processing Order (MAKINO)	Processing Hours
J1	9:00	16:00	8-1 $\Longrightarrow$ 8-2 $\Longrightarrow$ 8-1 $\Longrightarrow$ 8-3	1 ; 2 ; 1 ; 2
J2	9:00	16:00	12-2 $\Longrightarrow$ 12-3 $\Longrightarrow$ 12-1	2 ; 2 ; 1
J3	10:00	17:00	4	3
J4	8:00	17:00	20-2 $\Longrightarrow$ 20-1 $\Longrightarrow$ 20-2	2 ; 2 ; 2
J5	9:00	17:00	8-1 $\Longrightarrow$ 8-2 $\Longrightarrow$ 8-2 $\Longrightarrow$ 8-1	1 ; 1 ; 1 ; 1

Table 3. Job precedences (real case).

Job	J1	J2	J3	J4	J5
J1	*None*	False	False	False	**True**
J2	False	*None*	False	False	False
J3	False	False	*None*	False	False
J4	False	**True**	False	*None*	False
J5	False	False	False	False	*None*

It was essential to ensure that all MAKINO machines were supervised by workers during their operation, primarily due to safety concerns, quality assurance, and operational reliability. Continuous supervision reduces the risk of accidents, allows for immediate intervention in case of machine malfunction or unexpected errors, and ensures that the machining tasks meet the desired quality standards. The factory employee bank is full of workers available and capable of managing different types of MAKINO machines. Those employees can work at one of two available shifts. Initially, four workers were assigned to the industrial jobs related to the MAKINO machines. Details about each assigned employee are described in Table 4.

Table 4. Workers information (fictitious names).

ID	Name	Work Shifts
W1	João M.	8:00–17:00 (lunch break: 12:00–13:00)
W2	Ana S.	15:00–24:00 (dinner break: 19:00–20:00)
W3	Tiago R.	8:00–17:00 (lunch break: 12:00–13:00)
W4	Inês C.	8:00–17:00 (lunch break: 12:00–13:00)

Not all the previous workers have the same skills, as they have different training levels with the MAKINO machines, which are listed in Table 5.

Table 5. Worker skills (real case).

Worker/Machine	A4	A8.H	A12	A20.H
W1 (João M.)	Yes	Yes	No	No
W2 (Ana S.)	No	No	Yes	Yes
W3 (Tiago R.)	No	Yes	Yes	No
W4 (Inês C.)	Yes	No	Yes	Yes

When testing, it was verified that it is impossible to create a schedule with all the given data. Unfortunately, this can happen in real industrial contexts, as, in production environments, the reachability of those goals is very complex, which surpasses simple analysis.

This means that the original given data and constraints provided by the company are unreachable. The need for changes in the production goals is urgent, which is only possible by making some alterations to the source Excel file. But, before making considerable changes, it was necessary to discover which data makes this problem infeasible, by taking advantage of the selection options menu.

While selecting which set of conditions allows the program to turn the infeasible problem into a feasible one (by trial and error), it was found that only when the defined due hours and workers were turned off, the program generated a schedule for the MAKINO machines. This means that the necessary changes on the source data had to be made on the job due hours and/or assigned workers.

Alternative Solutions

Three alternative solutions were identified from making direct changes to the due hours and workers from the data source file, which now allows the generation of schedules for workers.

The first alternative solution resulted in the change of some job due hours, as shown in Table 6, and the addition of an extra worker (Sara F.) from the factory with the specifications from Table 7.

Table 6. Changes on some job due hours—for the 1st alternative solution.

Jobs	New Due Hour
J2, J5	21:00

Table 7. Additional worker—for the 1st alternative solution.

Added Worker	Work Shift	Skills
Sara F.	15:00–24:00 (dinner break: 19:00–20:00)	MA8

These modifications enabled the generation of the schedules illustrated in Figures 2 and 3.

However, this is not the only possible solution to surpass the blocks in the original data. In real industrial settings, it is common practice to reassign existing workers to different shifts than those originally scheduled, rather than hiring additional personnel to cover remaining hours. This approach offers greater flexibility, reduces labor costs, and makes more efficient use of the current workforce, especially when only a small number of additional work hours need to be covered. By adjusting shift assignments instead of increasing headcount, companies can respond more dynamically to production demands while maintaining operational efficiency.

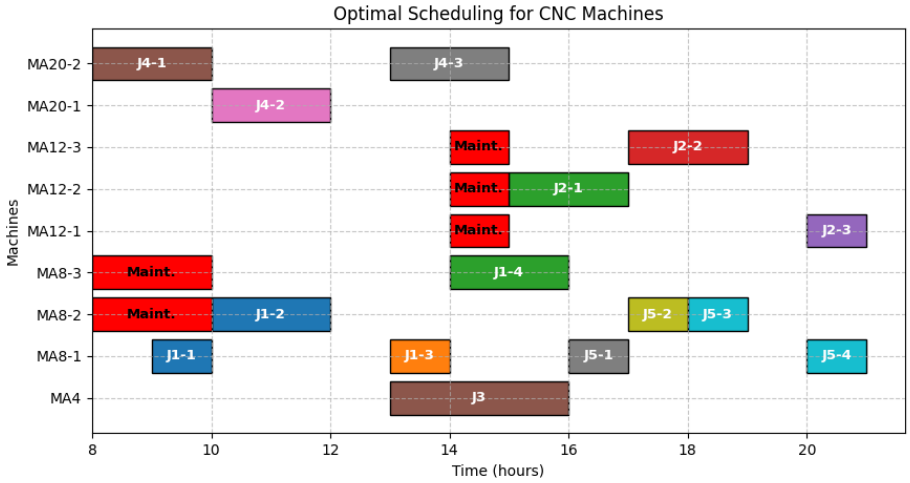


Figure 2. Optimal schedule for the MAKINO machines, taking into account the new due hours.

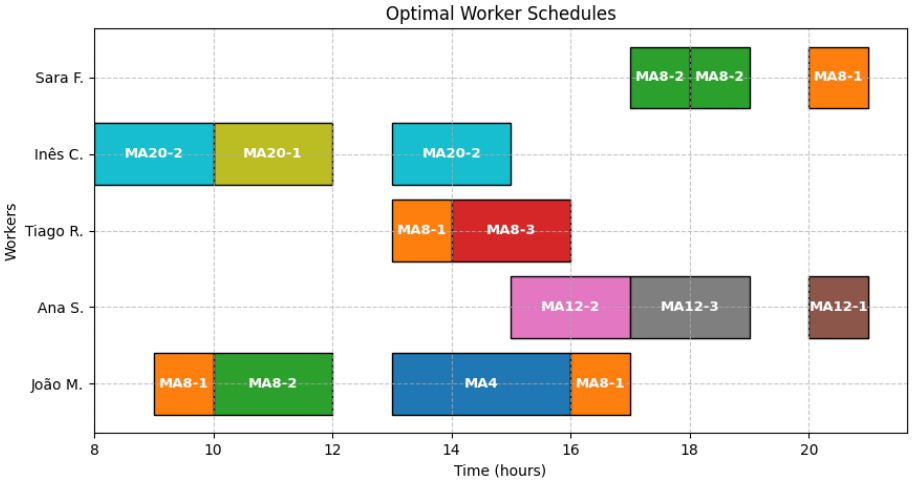


Figure 3. Optimal schedule for the workers, taking into account the added employee Sara F.

All the operations assigned to Sara F. on the first alternative solution were executed on machine MA8. One worker besides Sara F. who can manipulate this type of machine is Tiago R.

Though Tiago R. was assigned to work during the day shift (8:00–17:00), it can be observed on the first obtained alternative solution (Figure 3) that he starts to supervise machines at only 13:00, 5 h after his starting hour (8:00). Thus, asking him if he is willing to change his shift to the night one can be beneficial for the company, as it brings the opportunity to create schedules that do not need to include more workers beyond the ones originally planned.

The new working hours for Tiago R. are described in Table 8, and the respective changes on due hours are represenetd in Table 9.

Table 8. Tiago R. shift changes—for the 2nd alternative solution.

Worker	New Work Schedule
Tiago R.	15:00–24:00 (lunch break: 19:00–20:00)

Table 9. Changes on some job due hours—for the 2nd alternative solution.

Job	J1	J2	J5
New due hour	17:00	21:00	22:00

With the previous new information, two new schedules were generated by the program for the MAKINO machines and their assigned workers (Figures 4 and 5).

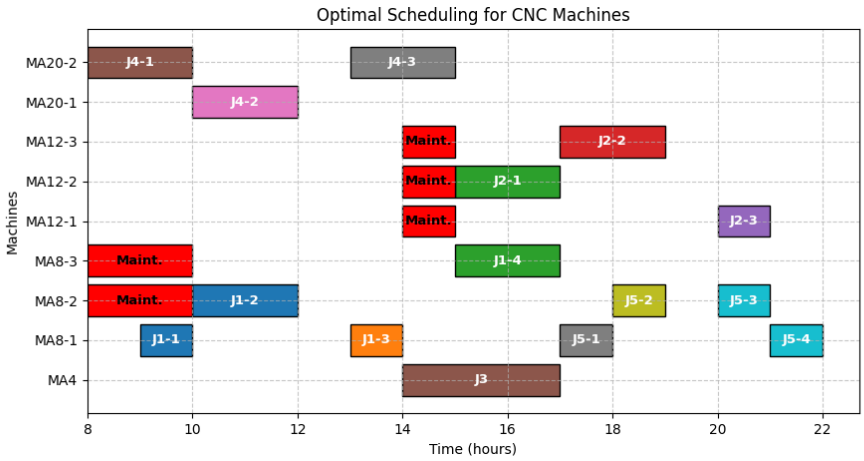


Figure 4. Optimal machine schedule, with Tiago R. working at the night shift.

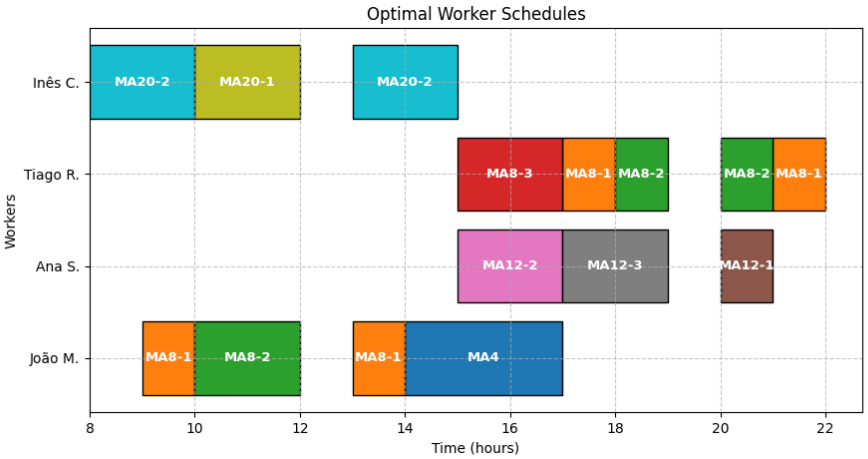


Figure 5. Optimal schedule for the workers, including Tiago R. working at the night shift.

The last option would be to provide extensive professional training to specific employees, so they become capable of handling more types of machines and, consequently, make the management of machine and worker schedules more flexible, without compromising the initial constraints imposed by the company. In the actual case study, originally, Ana S. could only supervise operations from MAKINO machines MA12 and MA20. Now, with some training, she can also manipulate MA8 machines (her new skills after her training are summarized in Table 10 and the new job due hours in Table 11).

Table 10. Ana S. skills to manipulate MA8 machines—for the 3rd alternative solution.

Machine	**MA4**	**MA8**	**MA12**	**MA20**
Has Ana S. the skill to manipulate it?	No	*Yes*	Yes	Yes

Table 11. Changes on some job due hours—for the 3rd alternative solution.

Job	**J2**	**J5**
New due hour	24:00	21:00

This small change was enough to generate the optimized schedules from Figures 6 and 7.

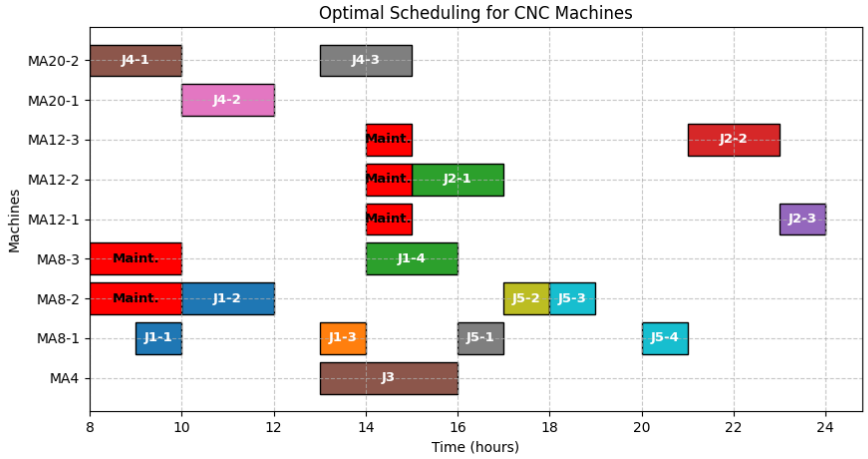


Figure 6. Optimal machine schedule, when given training to Ana S. to be able to manipulate machines MA8.

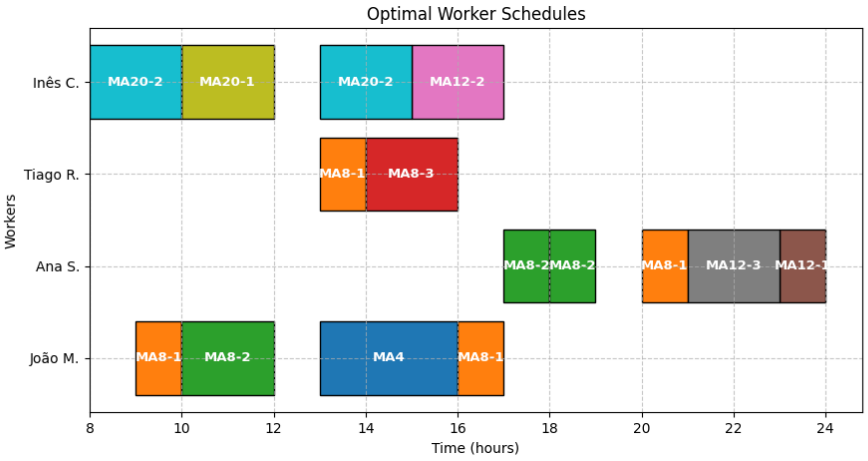


Figure 7. Optimal schedule for the workers, when given training to Ana S. to be able to manipulate machines MA8.

Even if the data is well defined, there is a risk that the requirements stated by the company may be impossible to achieve. The use of the switch on/off functionality in the developed software system proved to be highly useful for identifying, through a trial-and-error approach, which specific data inputs or constraints were preventing the solver from finding optimal solutions. By selectively enabling or disabling certain data inputs and/or constraints, users could isolate the problematic components, better understand the influence of each constraint, and iteratively refine the configuration to obtain a feasible solution and to improve the solution quality. From small changes on the original data source, it was possible to generate three different alternative optimal solutions that are summarized in Table 12.

Table 12. Obtained total makespans for the three alternative solutions, with the main applied changes and execution times.

Alternative Solution	***First***	***Second***	***Third***
Obtained makespan	117.0	124.0	124.0
Main change	Sara F. Added	Night shift for Tiago R.	Training for Ana S.
CBC CPU time (s)	0.18	0.15	0.14
HiGHS CPU time (s)	0.10	0.09	0.09

Both solvers (CBC and HiGHS) produced the same results for the three alternative cases in similar computational CPU execution times. The three shown alternative solutions are all capable of minimizing efficiently the total makespan; but only the first alternative solution obtained the lowest total makespan (117.0). However, this does not mean that this is the most desirable solution for the factory. Choosing the best solution for production organization and scheduling depends on the company's preferences and current conditions. The primary purpose of this system is to assist users in identifying the best possible solution based on the provided data and requirements. Rather than replacing human judgment, the system serves as a decision-support tool, helping users explore alternative scheduling scenarios, evaluate trade-offs between conflicting constraints, and make informed, data-driven decisions. By offering visual feedback and optimized solutions, the system empowers users, especially those without technical expertise in optimization, to effectively manage resources, improve operational efficiency, and adapt to changing production requirements.

6. Conclusions and Future Work

The main objective of evaluating the creation of an analytical autonomous decision-making software for the Portuguese factory was achieved successfully. Through the real-life case from a Portuguese industrial company, it was possible to show how well the developed software optimizes industrial CNC scheduling problems and offers a useful tool for actual industrial operations, like the previously referred factory. This type of no-code tools becomes more important as many of the employees that are responsible for assigning tasks to the available machines and workers in large-scale environments do not have the programming skills to write lines of code with MILP formulation constructor interfaces and optimization solvers, showing once again the importance of the combination of software with mathematical techniques in the industrial world. This aligns well with the core principles of Industry 4.0 and smart scheduling, where flexibility, automation, and decision-support are critical, particularly in environments where cyber-physical systems are fundamental and real-time dynamic scheduling is essential.

Although the developed software is capable of solving the scheduling problems currently specified by the factory in Portugal, it has the potential to be further enhanced to address a wider range of industrial needs. Future upgrades could incorporate improvements to the objective function to account not only for makespan but also for machine idle time and job delays, reflecting whether deadlines are met; besides additional data types and constraints, such as weekly or rotating shift schedules, flexible machine assignments, variable processing times, worker availability or absences, machine setup times, and raw material availability. Integrating these elements would significantly increase the versatility of this system, enabling it to handle more complex and dynamic scheduling environments. Such enhancements would make the software a more powerful and general-purpose decision-support tool, adaptable to diverse manufacturing contexts and capable of helping industrial managers optimize operations under more realistic and demanding conditions.

The usability of the GUI has not yet been formally evaluated, leaving user experience and accessibility aspects uncertain. To address this, future usability testing will involve employees from the Portuguese factory. Additionally, further tests incorporating a larger number of machines and jobs will be carried out to assess the system's scalability in more complex manufacturing environments.

One potential future improvement to the developed software is by adding a functionality that identifies which constraint(s) cause the problem to become infeasible, activated only when infeasibility occurs. This way, it becomes much easier for the user to identify the data and/or constraints that could be turned on or off or even changed, in order to transform the problem into a feasible one. Another interesting extension that could be

implemented in the near future is the possibility to change the original input data inside the program, without having to make modifications to the original Excel file (this could be useful in situations like the real-life case, where changes in the original data are mandatory to generate any schedule charts). Future work will apply more advanced algorithms, like genetic algorithms and dynamic programming, instead of the used MILP solvers, to study their effectiveness on solving CNC scheduling problems.

Author Contributions: Conceptualization, G.S.S.M., M.F.P.C. and F.A.; Methodology, G.S.S.M., M.F.P.C. and F.A.; Software, G.S.S.M. and F.A.; Validation, G.S.S.M., M.F.P.C. and F.A.; Formal analysis, G.S.S.M., M.F.P.C. and F.A.; Investigation, G.S.S.M., M.F.P.C. and F.A.; Writing—original draft, G.S.S.M.; Writing—review & editing, M.F.P.C. and F.A.; Supervision, M.F.P.C. and F.A.; Project administration, M.F.P.C. and F.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are unavailable due to privacy restrictions.

Acknowledgments: This work was partially financed by Portuguese Funds through FCT (Fundação para a Ciência e a Tecnologia) within the projects UIDB/00013/2020 through the BI scholarship 25/ECUM/CMAT/2024—UIDB/00013/2020 and UID/00013: Centro de Matemática da Universidade do Minho (CMAT/UM). This work was supported under the base funding project of the DTx—Digital Transformation CoLAB, project Missão Interface of the Recovery and Resilience Plan (PRR), integrated in the notice 01/C05-i02/2022, which aims to deepen the effort to expand and consolidate the network of interface institutions between the academic, scientific, and technological system and the Portuguese business fabric.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Appendix A.1. Notation

Table A1. Table of *mandatory* attributes and decision variables/parameters.

Classification	Type	Notation	Description
Attributes	Set	$\mathcal{M}$	*set of machine* unique identifications
	Set	$\mathcal{J}$	*set of job* unique identifications
	Continuous	p_{ij}	*processing time* of job j on machine i
	Integer	t_0	initial time value
	Set	T	set with all the created discrete time values
	Integer	T_{max}	time horizon
Aux. variable	Continuous	S_{ij}	*starting time* for job j on machine i
	Continuous	C_{ij}	*completion time* for job j on machine i
	Binary	ao_{ijt}	job j on machine i is still *active* at time t
Decision var. /parameters	Binary	x_{ijt}	machine i starts to process job j at time t
	Continuous	C_j	*makespan* of job j

Table A2. Table of *non-mandatory* attributes and decision variables/parameters.

Classification	Type	Notation	Description
Decision var. /parameter	Binary	y_{ikt}	worker k starts task from job j on machine i at discrete time t
Attributes	Continuous	P_j	*priority* of job j
	Continuous	r_j	*release date* of job j
	Continuous	d_j	*due date* of job j
	Set	$\mathcal{W}$	*set of worker* unique identifications
	Binary	$skill_{ki}$	worker k has the *skill* to operate machine i
	Binary	$U_{jj'}$	*job precedences*
	Integers	$\mathcal{O}_j$	*processing order* of job j
	Continuous	WS_k	*working schedules* of worker k
	Continuous	MS_i	*maintenance schedules* for machine i
Aux. variable	Binary	aw_{kit}	worker k working on machine i at time t

Appendix A.2. Screenshots from the Developed System

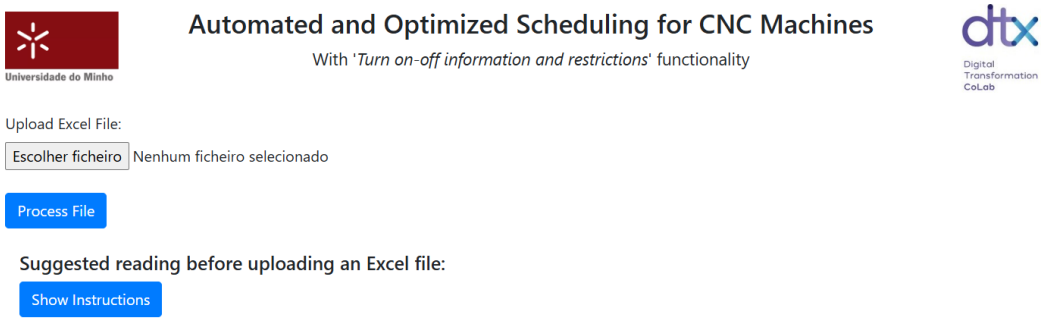


Figure A1. Startup/Upload page without instructions for filling an Excel file.

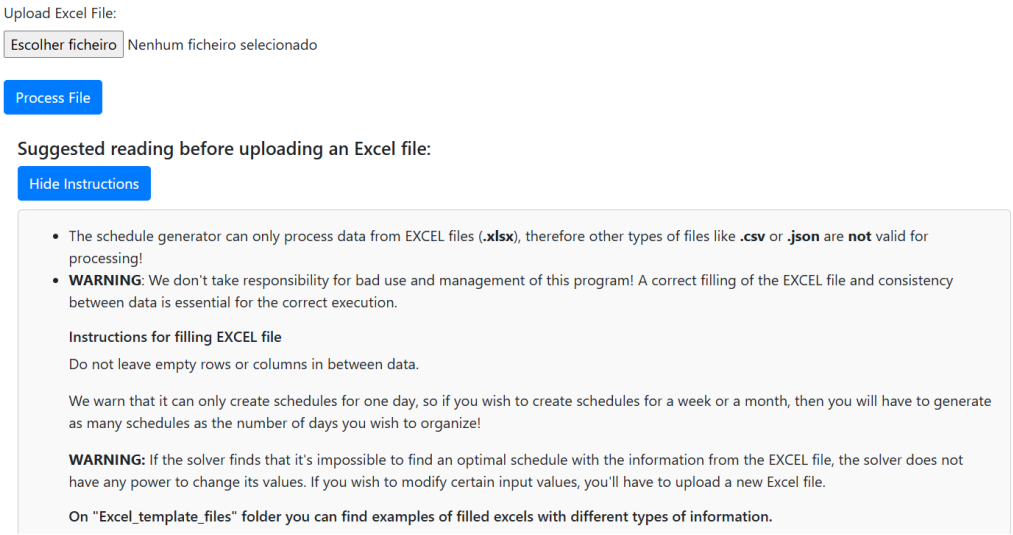


Figure A2. (Excerpt) Startup/Upload page when instructions for filling an Excel file are (partially) shown.

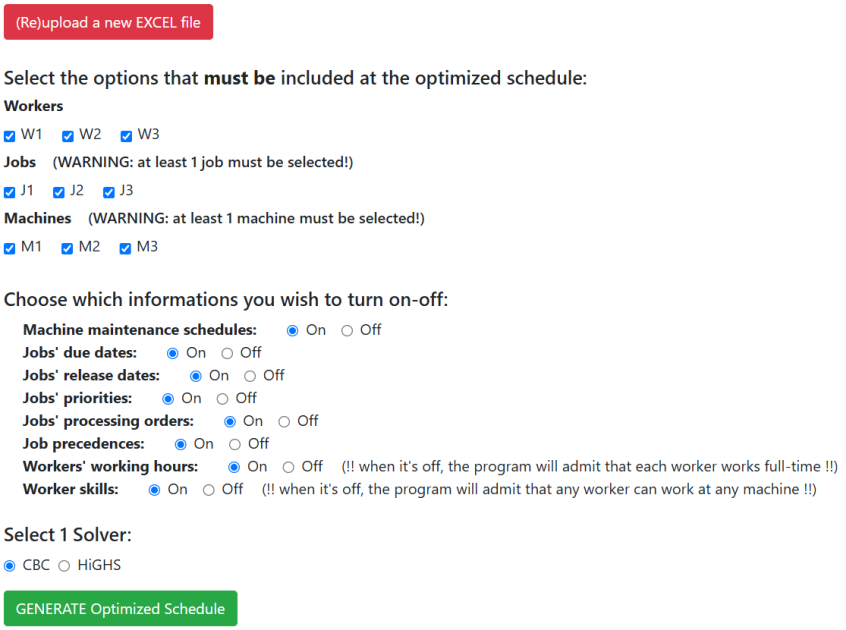


Figure A3. Option Selection page with the buttons to choose the information for scheduling.

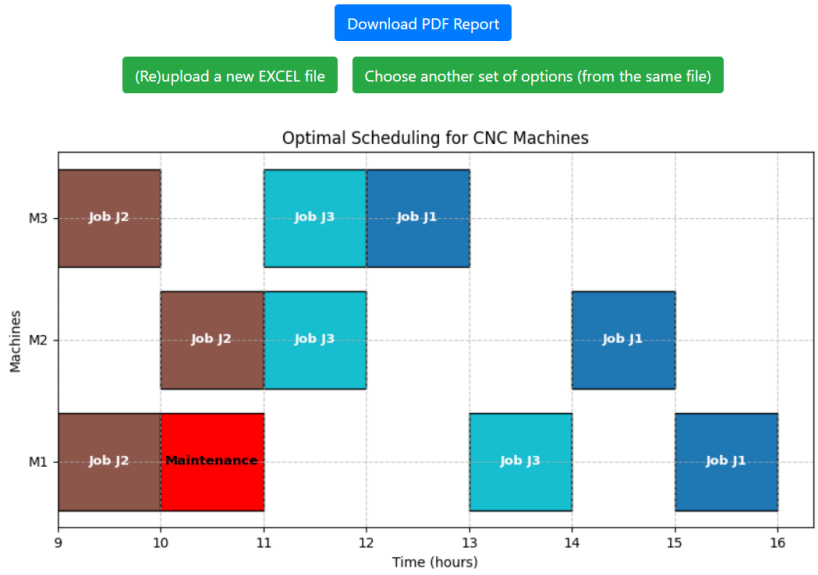


Figure A4. Results page with a fully created schedule.

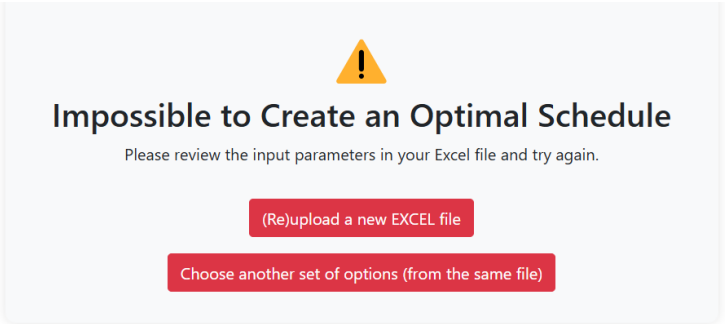


Figure A5. Error page with a warning message.

References

1. Google-Developers. Scheduling. 2024. Available online: https://developers.google.com/optimization/scheduling (accessed on 20 September 2024).

2. Google-Developers. Job Shop Scheduling. 2024. Available online: https://developers.google.com/optimization/scheduling/job_shop (accessed on 20 September 2024).
3. Xiong, H.; Shi, S.; Ren, D.; Hu, J. A survey of job shop scheduling problem: The types and models. *Comput. Oper. Res.* **2022**, *142*, 105731. [CrossRef]
4. CNC-Machines. What is a CNC Machine? 2024. Available online: https://cncmachines.com/what-is-a-cnc-machine (accessed on 23 September 2024).
5. Pinedo, M.L. *Scheduling: Theory, Algorithms, and Systems*, 6th ed.; Springer: Cham, Switzerland, 2022. [CrossRef]
6. Baker, K.R.; Trietsch, D. *Principles of Sequencing and Scheduling*, 2nd ed.; Wiley Series in Operations Research and Management Science; Wiley: Hoboken, NJ, USA, 2013.
7. Brucker, P. *Scheduling Algorithms*; Springer: Berlin/Heidelberg, Germany, 2007.
8. Zhang, J.; Liu, C.; Li, X.; Zhen, H.L.; Yuan, M.; Li, Y.; Yan, J. A survey for solving mixed integer programming via machine learning. *Neurocomputing* **2023**, *519*, 205–217. [CrossRef]
9. CBC (Coin-or Branch and Cut)—Introduction. Available online: https://coin-or.github.io/Cbc/intro.html (accessed on 22 November 2024).
10. Team, H.D. HiGHS Solvers Documentation. Available online: https://ergo-code.github.io/HiGHS/dev/solvers/#solvers (accessed on 21 November 2024).
11. Developers, P.M. Modeling Examples. 2024. Available online: https://python-mip.readthedocs.io/en/latest/examples.html (accessed on 26 September 2024).
12. Huangfu, Q.; Hall, J.A.J. Parallelizing the dual revised simplex method. *Math. Program. Comput.* **2018**, *10*, 119–142. [CrossRef]
13. Mahajan, A.; Hall, J.; Wong, E. Implementation of an interior point method with basis preconditioning. *Math. Program. Comput.* **2020**, *12*, 603–635. [CrossRef]
14. Wang, H.; Pan, Y.; Cui, W. Quantum-Inspired Solvers on Mixed-Integer Linear Programming Problem. *arXiv* **2022**, arXiv:2204.04574.
15. Manne, A.S. On the Job-Shop Scheduling Problem. *Oper. Res.* **1960**, *8*, 219–223. [CrossRef]
16. Nemhauser, G.L.; Wolsey, L.A. *Integer and Combinatorial Optimization*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 1988. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.