Mixed-Integer Linear Programming for Production Scheduling with Robot Allocation in a Textile Company

Maria Coelho Lima[1(✉)], Carina Pimentel[1,2], and Filipe Alvelos[1,2]

[1] ALGORITMI Research Center/LASI, School of Engineering, University of Minho, Campus of Azurém, 4800-058 Guimarães, Portugal
maria.afonso.coelho.lima@gmail.com

[2] Department of Production and Systems, School of Engineering, University of Minho, Campus of Azurém, 4800-058 Guimarães, Portugal

Abstract. This article addresses production scheduling challenges in the luxury textile industry, with a particular focus on robot-tufted carpet manufacturing. The study was developed in collaboration with a Portuguese consultancy company and targets the complexity arising from high product customization, strict delivery deadlines, and limited production resources.

A production scheduling model based on Mixed-Integer Linear Programming (MILP) is proposed to optimize the allocation and sequencing of carpets in the main robot-assisted production task. The model incorporates key operational constraints, including robot compatibility, yarn availability, and production deadlines. The objective is to minimize critical delays of the task related to completion within two days of the delivery date, promote completion at least eight days in advance, minimize makespan per robot production type, and enhance production anticipation while maintaining schedule feasibility. To guarantee computational efficiency in an industrial setting, solver parameter tuning strategies were implemented. To further improve performance, the Fix-and-Optimize heuristic was tested; however, its impact was limited for the problem scale considered. Consequently, a warm-start strategy based on existing production plans was adopted to provide high-quality initial solutions and enhance solver convergence.

The proposed approach was validated using real industrial data and benchmarked against current scheduling practices. Results show that optimized schedules increased the proportion of orders completed at least two days before delivery by 1.9% and at least eight days before by 3.7%, while also reducing makespan.

Overall, the study demonstrates the effectiveness of mathematical optimization as a decision-support tool for complex production environments, reducing scheduling time while improving service levels and operational performance in the luxury manufacturing sector.

Keywords: Textile Industry · Production Scheduling · Mixed-Integer Linear Programming

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026 Workshops, LNCS 16760, pp. 306–323, 2027.
https://doi.org/10.1007/978-3-032-30521-3_20

1 Introduction

The textile sector is dynamic and subject to constant market fluctuations, making it essential to adopt advanced planning techniques that improve both responsiveness and decision quality [1]. At the same time, increasing product customization raises production variability and complexity in planning and forecasting. This scenario calls for the integration of advanced data-driven approaches [2].

This work is framed within a Portuguese textile company dedicated to the production of luxury carpets, operating two manufacturing units. The company follows a made-to-order strategy, producing both catalog and bespoke products. Approximately 70% of orders correspond to customizable catalog models, while the remaining 30% are fully bespoke, requiring flexible and responsive production planning.

The company relies on four main production techniques: flatweave, hand-knotted, hand-woven, and robot-tufted. Among these, robot-tufted is the most technologically advanced and operationally significant, representing the majority of current output. This process relies on automated robotic systems to inject yarn into a stretched carpet base according to predefined digital trajectories.

The robot-tufted process involves multiple departments, including order processing, design, yarn availability, and scheduling. Currently, scheduling is manual and relies on planners' experience, often leading to inefficiencies and an average delay rate of around 12%. These challenges motivated the development of an optimization-based decision-support model capable of generating sequenced production plans based on the order portfolio and operational constraints. In particular, this work proposes a Mixed-Integer Linear Programming model that integrates allocation and sequencing decisions while incorporating key industrial constraints, and evaluates its performance using real production data.

The remainder of this paper is organized as follows. Section 2 presents background on production scheduling. Section 3 introduces the proposed model. Section 4 describes the solution approach. Section 5 presents the results, and Sect. 6 concludes the paper.

2 Theoretical Background

Production scheduling is a decision-making process used in manufacturing and service sectors to allocate resources such as labor, machines, or capital to tasks in a defined sequence and over time [3].

Historically, scheduling was considered a simple step following production planning. However, modern manufacturing environments, characterized by high variability, customization, and complex resource interactions, require more advanced scheduling methodologies. Advances in decision support optimization make it increasingly feasible to replace manual spreadsheet-based planning with robust analytical tools capable of generating highly efficient solutions [4]. These methods help identify near-optimal solutions that minimize production time and improve resource utilization [5].

This section presents an overview of production scheduling, including problem classification, solution approaches, and computational efficiency strategies, followed by relevant applications in manufacturing contexts.

2.1 Classification of Scheduling Problems

Scheduling problems are commonly described using the three-field notation $\alpha|\beta|\gamma$ introduced in [7].

The first element, α, defines the machine environment, influencing problem complexity and enabling comparison between scheduling approaches. Common environments include single-machine, parallel-machine, and multi-machine shop systems.

Single-machine environments process tasks sequentially on one resource, while parallel-machine environments allow tasks to be processed by multiple machines, which may be identical, uniform, or unrelated. Multi-machine shop environments include flow shops, job shops, and open shops, each characterized by different routing and sequencing structures.

The second field, β, describes attributes and constraints associated with jobs, such as precedence relations, release dates, setup times, operation limits, and unit processing time assumptions. These features influence model complexity and solution choice.

The third field, γ, defines the performance objective used to evaluate a schedule, such as makespan, lateness, tardiness, earliness, setup time, and weighted objectives reflecting job priorities.

2.2 Scheduling Approaches

Since the mid-1950s, scheduling problems have increased in number and complexity [8]. This has led to a range of approaches, from simple heuristics to advanced mathematical models.

A common approach uses priority-based dispatching rules to select the next task for processing. Dispatching rules have a significant role within the dynamic context due to their ease of implementation and compatibility with the dynamic nature of manufacturing systems [9]. Typical examples include First-Come, First-Served, Earliest Due Date, Shortest or Longest Processing Time, and the Minimum Slack rule.

Mathematical programming, particularly Mixed-Integer Linear Programming, has become one of the most widely explored methods for process scheduling due to its rigor, flexibility, and strong modeling capabilities [10]. Scheduling models can use discrete or continuous time, with continuous formulations often preferred due to lower approximation errors and computational burden.

Continuous-time models can be further categorized as immediate precedence or general precedence formulations. Immediate precedence models enforce that one task starts directly after another on the same machine [11], whereas general precedence formulations impose only relative ordering constraints between

tasks [12]. A comparative study found that the general precedence formulation is the most computationally efficient among the commonly used models [13], motivating its adoption in this research work.

Constraint Programming (CP) is a declarative paradigm for solving combinatorial problems by explicitly defining constraints that feasible solutions must satisfy [14]. These approaches are valued for their modeling flexibility and integration in industrial decision-support systems.

Finally, Metaheuristics balance exploration and exploitation to obtain high-quality solutions efficiently [15]. Common methods include Simulated Annealing, Genetic Algorithms, and Tabu Search, which are effective for complex problems despite not guaranteeing optimality.

2.3 Computational Efficiency and Applications

Scheduling problems are computationally complex, making efficiency critical in industrial applications. Performance can be improved through model simplification, decomposition, warm starts, and solver tuning.

Model simplification reduces problem size by relaxing non-critical constraints or aggregating variables. Decomposition methods divide large problems into smaller subproblems, for example, Fix-and-Optimize iteratively fixes subsets of variables and re-optimizes the remaining ones [16]. Warm-start techniques provide an initial feasible solution to guide the branch-and-bound search process. Solver parameter tuning can also significantly affect performance [17].

Optimization-based approaches have been applied to textile and related manufacturing industries. A MILP model for production planning in a luxury textile company was presented in [18], incorporating machine capacity and material constraints. A weighted-tardiness problem in a textile manufacturing context was addressed using a hybrid MILP and metaheuristic approach in [19]. Earlier work described scheduling challenges in carpet production within traditional non-automated environments [20].

While prior studies highlight the relevance of mathematical programming in textile manufacturing, limited work addresses automated carpet production with jointly coordinated robotic resources, material availability, and delivery constraints. This work fills that gap with an integrated MILP model for robot-assisted production.

3 Optimization Modeling

The robot-tufted production stage begins with robotic yarn injection and is followed by finishing operations, which include trimming, and quality assurance steps. Since the first stage determines the start of the subsequent processes, the scheduling model developed in this work focuses specifically on this stage.

The operational rules governing the injection stage were first mapped and then formalized as a MILP model. In this formulation, products are assigned and sequenced on compatible robots according to their typology (injection type).

Structure allocation depends solely on dimensional compatibility, with a mandatory 20 cm safety margin. Production can start only two days after the latest yarn arrival and must respect the working calendar, labor capacity, planned stoppages, and setup requirements. The first two days of the schedule are kept fixed.

Special cases, such as carpets that are too small or too large to follow the standard rules, introduce additional timing and assignment constraints. No explicit planning horizon is imposed, as all pending orders must be scheduled.

Based on these constraints and planning needs, the model was designed with the following objectives:

- Minimize the number of delays, where a minor delay is defined as an injection stage ending less than eight days before the delivery date, reducing the buffer time for finishing and shipping, and a major delay occurs when injection ends less than two days before the delivery date, making it almost impossible to meet the delivery deadline;
- Minimize the time each robot typology takes to complete all assigned orders;
- Minimize the time lag between material availability and the initiation of production.

While the first two objectives were included in the initial formulation, the third was introduced after preliminary tests revealed idle gaps despite material availability.

3.1 Data Preparation

To simplify the formulation and reduce computational complexity, several preparatory steps were applied before modeling. Instead of incorporating all conditions directly as constraints, part of the filtering and structuring was conducted in advance through data preparation.

The first filtering step selected unplanned products that had already undergone color analysis, ensuring yarn requirements and availability were known.

Robot information was then collected, including typology, dimensional capacity, manufacturing unit, and activity status, retaining only active robots. A similar procedure was applied to structures, considering size, activity status, and manufacturing unit. The company's manufacturing calendar was also processed to obtain the start and end times of planned robot stoppages.

Product data were consolidated from multiple sources, starting with the verification of yarn availability. For products without yarn in stock or a defined arrival date, an artificial arrival time was assigned by the company. This was set to five days from the current date if the delivery date was within twenty-one days, and to twelve days otherwise. This approach allowed these products to be scheduled while maintaining a realistic buffer for yarn availability. For each product, the latest arrival among all required yarns was selected, and two additional days were added to account for yarn preparation.

Products were also assigned a typology based on the robot previously allocated by planners. Although this assignment was necessary, future work should define typology independently to avoid reliance on initial plans. Production duration was then calculated by dividing the effective area by the injection rate estimated by planners, ensuring consistency with company practice.

A key methodological decision was the adoption of continuous time rather than indexed time periods. This choice increases the realism and flexibility of the formulation by allowing tasks to be scheduled with greater precision. This choice is supported by the characteristics of the robot-tufted process. Robots operate autonomously during production and require human intervention only during setup operations. As a result, production can continue uninterrupted even during worker breaks. All planned machine stoppages are known in advance and incorporated into the model as scheduled interruptions. All relevant dates (delivery deadlines, yarn arrivals, and stoppages) were converted into hours measured from the start of the planning horizon (hour zero).

Building on this preprocessing, several datasets were created to encode feasibility conditions, ensuring that only feasible allocations and sequencing possibilities were considered by the model. These datasets form the basis of the mathematical formulation and are presented next.

- PR: all feasible product–robot combinations, filtered by compatibility of typology and dimensions, as well as manufacturing unit restrictions. For Manufacturing Unit 2, only products with all yarns already in stock were included.
- PS: all feasible product–structure allocations, filtered by dimensions with a mandatory security margin of 20 cm on all borders, along with the same manufacturing unit restrictions.
- C: the set of all carpet samples, which must be assigned to a robot but not to a structure, since they are handled differently in production.
- PM: the set of product–manufacturing unit combinations, created only for products with yarn in stock, which can potentially be produced in either unit.
- PP' and $PP'R$: pairs of products with the same typology, and their feasible allocations to the same robot, respectively, supporting the sequencing constraints. During dataset creation, setup times between consecutive products were specified: fifteen minutes for standard machine preparation, reduced to five minutes for products from the same order since the same yarn is used. This differentiation encourages the model to schedule products from the same order consecutively on the same robot.
- F: products with fixed allocations for the first planning days, which the model cannot modify. The number of fixed days can be adjusted as required.
- PKR and PK: respectively, combinations of product–robot pairs (from PR) with stoppages, used to model robot-dependent interruptions, and combinations of products with stoppages, used for robot-independent interruptions.
- Special carpets are subdivided into three datasets:
  - SD: descending carpets are carpets too large to be produced in a single pass and must be partially produced, repositioned (descended), and then completed. In the data, they appear as two products with different map

codes but the same product code, and a dataset was created to link these pairs so the model treats them correctly.
- *SS*: spliced carpets are also large carpets but produced in two parts and later joined. They appear as separate products with different map and product codes, making them impossible to identify automatically. A dedicated dataset was therefore used to detect these pairs, which were then consolidated into a simpler dataset for the model.
- *SSS*: splice cases where the first half was previously produced and the second must now be scheduled, typically as rework. This dataset retrieves the splice information and the robot used for the first half, ensuring the second half is produced on the same robot.

In addition, carpets small enough to be grouped were treated as a single aggregated product with combined dimensions, since they are injected together in the same operation, without requiring a dedicated dataset.

In summary, the data preparation phase ensured model feasibility while keeping the formulation manageable by providing structured and accurate inputs.

3.2 Mathematical Formulation

The formulation is structured into indices and sets, parameters, decision variables, constraints, and an objective function.

Indices and Sets. Indices and sets provide the structural basis of the formulation: indices reference individual elements, while sets define the collection of feasible elements.

For this model, the basic indices and their respective sets are:

- $p \in P$: products
- $r \in R$: robots
- $t \in T$: injection typologies
- $s \in S$: structures
- $m \in M$: manufacturing units
- $k \in K$: planned stoppages

Additional feasibility datasets such as PR, PS, C, PM, PP', $PP'R$, F, PKR, PK, SD, SS, and SSS are incorporated into the formulation.

Parameters. Parameters define the input characteristics of the scheduling problem, describing the attributes of products, resources, and stoppages. All time-related parameters were expressed in hours for consistency.

- $Duration_p$: production time for product p
- $Typology_p, Typology_r$: typology of product p and robot r
- $Delivery_p$: delivery deadline of product p

- $Yarn_p$: time of the latest yarn arrival for product p
- $Setup_{pp'}$: setup time between products p and p', depending on order relation
- $KStart_k, SEnd_k$: start and end times of stoppages k
- $KDuration_k$: duration of stoppage k
- $PlannedStart_p$: start time for product p in previous plan

Decision Variables. Decision variables represent the model's choices for building a feasible schedule, defining product allocation, sequencing, and operation timing. Both binary and continuous variables are used, with time expressed in hours.

$$X_{pr} = \begin{cases} 1 & \text{if product } p \text{ is assigned to robot } r \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$Y_{ps} = \begin{cases} 1 & \text{if product } p \text{ is assigned to structure } s \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$Z_{pp'} = \begin{cases} 1 & \text{if product } p \text{ precedes } p' \text{ on the same robot} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$W_{p,k} = \begin{cases} 1 & \text{if stoppage } k \text{ interrupts product } p \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$B_{p,k} = \begin{cases} 1 & \text{if product } p \text{ finishes before stoppage } k \text{ begins} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$Start_p$: start time of product production p.

End_p: end time of product production p.

$Makespan_t$: completion time of the last product for each injection typology t.

$Tardiness2_p$: delay recorded when the production of product p ends less than two days before its delivery date.

$Tardiness8_p$: delay recorded when the production of product p ends less than eight days before its delivery date.

$StartAfterYarn_p$: time between yarn availability and start of product p.

Constraints. Constraints define the operational and logical rules that any feasible solution must satisfy, ensuring the production plan respects resource, timing, and sequencing requirements.

Allocation constraints, presented below, ensure each product is assigned to available and appropriate manufacturing resources.

$$\sum_{r \in R:(p,r) \in PR} X_{pr} = 1, \ \forall p \in P \tag{6}$$

$$\sum_{s \in S:(p,s) \in PS} Y_{ps} = 1, \ \forall p \in P \setminus C \tag{7}$$

$$\sum_{s \in S:(p,s) \in PS} Y_{ps} = 0, \ \forall p \in C \tag{8}$$

$$\sum_{r \in R:(p,r) \in PR, \ M(r)=m} X_{pr} = \sum_{s \in S:(p,s) \in PS, \ M(s)=m} Y_{ps}, \forall (p,m) \in PM \setminus C \tag{9}$$

First, each product must be assigned to exactly one robot, as defined in Constraint (6). Then, the products must also be assigned to exactly one structure, as specified in Constraint (7), with the exception of sample products, which cannot be allocated to any structure (Constraint (8)). Furthermore, each product must be assigned to both a robot and a structure within the same manufacturing unit, as enforced by Constraint (9). Since sample products are not assigned to any structure, this constraint does not apply to them.

Following this, production constraints ensure the temporal feasibility of the schedule, taking into account material availability and production duration.

$$Start_p \geq Yarn_p, \ \forall p \in P \tag{10}$$

$$End_p \geq Start_p + Duration_p + \sum_{k:(p,k) \in PK} (W_{p,k} \times KDuration_k), \ \forall p \in P \tag{11}$$

Initially, the production of each carpet can only start once the required yarn has arrived, as defined in Constraint (10). Additionally, the completion time of each carpet must be at least equal to its start time plus the production duration, including possible interruptions due to stops, as specified in Constraint (11).

Robot sequencing constraints ensure that two products assigned to the same robot ($X_{pr} = 1$ and $X_{p'r} = 1$) are produced sequentially without overlap, considering setup times. This is enforced through binary sequencing variables and a big-M constant, which exceeds any possible start or end time within the planning horizon.

$$Start_{p'} + M(1 - X_{pr}) + M(1 - X_{p'r}) + M(1 - Z_{pp'}) \geq End_p + Setup_{pp'}, \\ \forall (p, p', r) \in PP'R \tag{12}$$

$$Start_p + M(1 - X_{pr}) + M(1 - X_{p'r}) + MZ_{pp'} \geq End_{p'} + Setup_{p'p}, \\ \forall (p, p', r) \in PP'R \tag{13}$$

Constraint (12) ensures that if product p precedes product p' on the same robot, then the start of p' must occur after the end of p plus the corresponding setup time. Conversely, Constraint (13) enforces that if product p' precedes product p, then the start of p must not occur earlier than the end of p' plus its respective setup time.

Fixed production constraints preserve predetermined schedules for specific products during the first production days while allowing flexibility for the remaining schedule, ensuring that fixed assignments, start times, and sequencing are respected.

$$X_{pr} = 1, \ \forall (p, r) \in F \tag{14}$$

$$Start_p \geq PlannedStart_p, \ \forall r \in R, \ p = \text{first fixed product of } r \tag{15}$$

As a first step, products in the set of fixed assignments must be allocated to their predefined robot, as defined in Constraint (14). To account for products whose production starts before the beginning of the planning horizon and has not yet been completed, the first product scheduled on each robot must have a start time no earlier than its previously planned starting hours, as specified in Constraint (15).

$$Z_{pp'} \geq 1, \ \forall (p, p') \in PP', \ p, p' \in F : \ PlannedStart_p \leq PlannedStart_{p'} \tag{16}$$

$$Z_{pp'} = 0, \ \forall (p, p') \in PP', \ p, p' \in F : \ PlannedStart_p \geq PlannedStart_{p'} \tag{17}$$

The relative order of fixed productions must also be preserved. If a fixed production p is scheduled to start before another fixed production p', then the binary variable $Z_{pp'}$ is set to 1, as defined in Constraint (16). Conversely, if p is scheduled to start after p', then $Z_{pp'}$ must be equal to 0, as specified in Constraint (17).

$$Z_{pp'} + M(1 - X_{p'r}) \geq 1, \ \forall (p, p') \in PP', \ p \in F, \ p' \notin F \tag{18}$$

$$Z_{pp'} + M(1 - X_{p'r}) \leq 0, \ \forall (p, p') \in PP', \ p \notin F, \ p' \in F \tag{19}$$

Finally, to avoid conflicts between fixed and non-fixed productions assigned to the same robot ($X_{pr} = 1$), sequencing rules ensure that non-fixed productions are scheduled only after fixed ones. This requirement is enforced by Constraints (18) and (19).

Stoppage constraints ensure that products are scheduled correctly with respect to machine stoppages, considering whether they are interrupted, completed beforehand, or postponed. Binary variables $W_{p,par}$ and $B_{p,par}$ are used to model whether product p is interrupted by stoppage k or completed before it, respectively. For all these constraints, the product must be allocated to the robot to which the stoppage refers ($X_{pr} = 1$), enforced through the big-M approach.

$$Start_p \leq KStart_k + (1 - W_{p,k})M + (1 - X_{pr})M, \ \forall (p, k, r) \in PKR \tag{20}$$

$$Start_p + Duration_p + \sum_{\substack{(p,k',r) \in PKR: \\ KStart_{k'} < KStart_k}} (W_{p,k'} \times KDuration_{k'}) \geq KStart_k - (1 - W_{p,k})M - (1 - X_{pr})M, \forall (p,k,r) \in \text{PKR} \tag{21}$$

Initially, for a product to be interrupted by a stoppage ($W_{p,k} = 1$), it must start before the stoppage begins, as defined in Constraint (20). Furthermore, its production must extend beyond the start time of the stoppage, taking into account its duration and any previous stoppages that may have occurred, since a product can be interrupted by more than one stoppage, as specified in Constraint (21).

$$Start_p + Duration_p + \sum_{\substack{(p,k',r) \in PKR: \\ KStart_{k'} < KStart_k}} (W_{p,k'} \times KDuration_{k'}) \geq KStart_k + (1 - X_{pr})M + (1 - B_{p,k})M, \ \forall (p,k,r) \in \text{PKR} \tag{22}$$

If a product finishes before the stoppage begins, it is not interrupted. This case is modeled in Constraint (22), which activates when $B_{p,k} = 1$.

$$Start_p \geq KEnd_k - W_{p,k}M - (1 - X_{pr})M - B_{p,k}M, \ \forall (p,k,r) \in PKR \tag{23}$$

Similarly, if a product starts only after the stoppage has ended, it is not interrupted ($W_{p,k} = 0$) and is not completed before the stoppage ($B_{p,k} = 0$), as defined in Constraint (23).

$$\sum_{p \in P} W_{p,k} \leq 1, \ \forall k \in K \tag{24}$$

$$W_{p,k} + B_{p,k} \leq 1, \ \forall (p,k) \in PK \tag{25}$$

Lastly, to ensure consistency, at most one product can be interrupted by each stoppage, as defined in Constraint (24), and a product must either be interrupted by a stoppage or completed before it, but not both, as specified in Constraint (25).

Special carpet constraints ensure correct handling of carpets produced in two parts (descending or spliced), maintaining allocation consistency and temporal feasibility.

$$X_{p1r} = X_{p2r}, \forall (p1,p2) \in (SD \cup SS), \ \forall r \in R: \ (p1,r) \in PR, \ (p2,r) \in PR \tag{26}$$

$$Y_{p1s} = Y_{p2s}, \ \forall (p1,p2) \in SD, \ \forall s \in S: \ (p1,s) \in PS, \ (p2,s) \in PS \tag{27}$$

First, allocation consistency must be preserved. For descending carpets or carpets with splices, both parts must be assigned to the same robot, as defined in Constraint (26). Moreover, descending carpets must also be assigned to the same structure, as specified in Constraint (27).

$$Start_{p2} \geq Start_{p1} + Duration_{p1} + \sum_{k:\,(p1,k)\in PK} (W_{p1,k} \times KDuration_k) + 24, \quad \forall (p1, p2) \in SD \tag{28}$$

$$Start_{p2} \leq Start_{p1} + Duration_{p1} + \sum_{k:\,(p1,k)\in PK} (W_{p1,k} \times KDuration_k) + 32, \quad \forall (p1, p2) \in SD \tag{29}$$

In addition to allocation, temporal consistency must be ensured. For descending carpets, the second part must begin at least 24 h and at most 32 h after the first part is completed, as defined in Constraints (28) and (29).

$$Start_{p2} \geq Start_{p1} + Duration_{p1} + \textstyle\sum_{k:\,(p1,k)\in PK} (W_{p1,k} \times KDuration_k) + 0.0833, \quad \forall (p1, p2) \in SS \tag{30}$$

$$Start_{p2} \leq Start_{p1} + Duration_{p1} + \textstyle\sum_{k:\,(p1,k)\in PK} (W_{p1,k} \times KDuration_k) + 0.5, \quad \forall (p1, p2) \in SS \tag{31}$$

For splices, the second part must begin at least 5 min (0.0833 h) and at most 30 min (0.5 h) after the first part is completed, as defined in Constraints (30) and (31).

$$X_{pr} = 1, \ \forall (p, r) \in SSS \tag{32}$$

In addition, for singular splices where the first part has already been completed, the second part must be assigned to the same robot as the one previously used, as defined in Constraint (32).

Finally, the objective function uses auxiliary variables capturing tardiness, makespan, and non-productive gaps, defined by the following constraints.

$$Tardiness2_p \geq End_p - (Delivery_p - 48), \ \forall p \in P \tag{33}$$

$$Tardiness8_p \geq End_p - (Delivery_p - 192), \ \forall p \in P \tag{34}$$

To begin with, the tardiness variables capture whether the production of a carpet is completed close to its delivery deadline. Constraint (33) accounts for delays occurring within two days (48 h) before the delivery date, while Constraint (34) captures delays occurring within eight days (192 h).

$$Makespan_t \geq End_p,\ \forall p \in P \text{ such that } Typology_p = t,\ \forall t \in T \tag{35}$$

The makespan is defined for each typology $t \in T$, ensuring that it is no smaller than the completion time of any product p whose typology corresponds to t, as expressed in Constraint (35).

$$StartAfterYarn_p \geq Start_p - Yarn_p,\ \forall p \in P \tag{36}$$

As a final step, to measure the time between the yarn arrival and the production start, Constraint (36) is introduced.

Objective Function. The objective function aims to minimize the weighted sum of several decision variables that represent different dimensions of scheduling performance, as shown in Eq. (37). These include makespan, tardiness relative to the two- and eight-day delivery thresholds, and waiting time between yarn availability and production start. Together, they provide a comprehensive measure of the overall efficiency of the schedule.

$$\begin{aligned} \min \Big(& w_1 \times \textstyle\sum_{t \in T} Makespan + w_2 \times \sum_{p \in P} Tardiness2_p \\ & + w_3 \times \textstyle\sum_{p \in P} Tardiness8_p + w_4 \times \sum_{p \in P} StartAfterYarn_p \Big) \end{aligned} \tag{37}$$

Each component of the objective function is multiplied by a weight (w_1, w_2, w_3, w_4), that defines its relative importance. These weights can be adjusted by the company according to its operational priorities. Minimizing makespan shortens the production horizon, minimizing tardiness relative to the two- and eight-day thresholds improves delivery reliability, and minimizing the time between yarn availability and production start reduces idle periods and improves resource utilization, ultimately enhancing schedule reliability and responsiveness.

4 Model Solving

The optimization model was implemented in Python using the PuLP library with the Gurobi Optimizer (version 11.0.3), a solver known for efficiently handling large-scale MILP problems.

The model was implemented through Python functions that progressively built the formulation, improving transparency and maintainability, with logging and progress tracking used to monitor solver performance. Once all variables, constraints, and the objective function were defined, the model was solved using Gurobi. During testing, the model exhibited high complexity, with several hundred thousand variables and constraints, leading to substantial computational effort and longer solution times. As a result, specific strategies to improve computational efficiency had to be explored, in order to obtain feasible solutions within a reasonable time.

To address computational challenges, several strategies were explored. Despite Gurobi's efficiency, solving the model was challenging, with slow feasibility detection and gap reduction. The optimality gap measures the relative difference between the best feasible solution and the best known lower bound, indicating how close the solution is to optimality.

The first approach involved tuning Gurobi's solver parameters to improve feasibility detection and convergence. The configuration that yielded the best results is summarized in Table 1.

Table 1. Best-performing Gurobi parameter configuration

Parameter	Value	Definition
MIPFocus	1	Prioritizes finding feasible solutions over proving optimality.
Heuristics	1	Increases the use of heuristics to obtain initial solutions faster.
Cuts	2	Applies additional cuts to strengthen the formulation.
Threads	4	Parallel threads to speed up the generation of schedules.

This configuration improved feasibility detection, but the solver often stalled at a given optimality gap. To address this, a Fix-and-Optimize heuristic was tested, which fixes part of the decision variables while optimizing the others to reduce problem size. Although this approach helped identify feasible solutions more quickly, it often plateaued without further improving the optimality gap.

Given these limitations, a warm start strategy was developed and proved more effective. A temporary constraint (Constraint (38)) was introduced to enforce the allocation of the same robot previously assigned to each order, when it appeared in an earlier production plan.

$$X_{pr} = 1, \quad \forall p \in p_{\text{planned}},\ r = r(planned)_p \tag{38}$$

The model was first solved under these restrictions for about five minutes to obtain an initial feasible solution. The constraint was then removed and the model restarted using this solution as a warm start. The solver was run for one hour, achieving a single-digit optimality gap with a good runtime–quality trade-off. This approach produced an initial solution almost instantly and improved the gap frequently, unlike the Fix-and-Optimize approach, where improvements required several minutes.

5 Results, Discussion and Implementation

Several production plans were generated and compared with the company's plans using key performance metrics. Five plans were considered, each with a different initial planning date and order set, covering a four-week horizon corresponding to the company's standard planning window.

The plans were generated during validation sessions in which the planning team directly tested the model and compared it with the corresponding company plans. The objective function weights (37) were initially defined based on the company's priorities, resulting in $w_1 = 1$, $w_2 = 12$, $w_3 = 2$, and $w_4 = 0.75$. However, this configuration led to an excessive focus on minimizing Dd-2 tardiness at the expense of makespan, revealing the sensitivity of the model to weight calibration. The weights were therefore adjusted based on empirical evaluation to $w_1 = 2$, $w_2 = 6$, $w_3 = 2$, and $w_4 = 0.75$, accounting for the relative magnitude of the objective components and yielding more balanced trade-offs.

Performance was evaluated using the average percentage of products completed on time across the five plans, considering two thresholds: completion at least two days (Dd-2) and eight days (Dd-8) before the delivery date (Fig. 1).

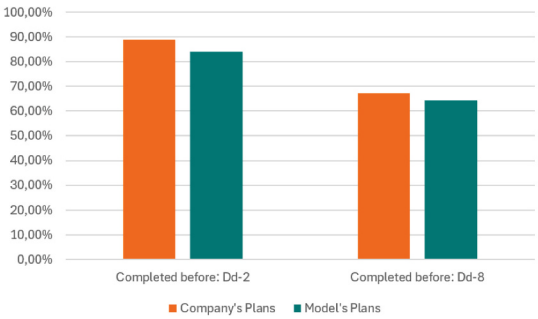

Fig. 1. Comparison of average tardiness (Dd-2 and Dd-8) between the company's plans and the model's plans

The model's plans initially underperformed compared with the company's plans in both thresholds, achieving 84.1% (Dd-2) and 64.3% (Dd-8), versus 88.9% and 67.1%, respectively. This gap was mainly driven by outdated yarn arrival data, which led to suboptimal scheduling decisions. In practice, planners often rely on informal updates regarding earlier yarn arrivals that are not reflected in the system, highlighting the importance of accurate yarn management and data updates.

To address this issue, yarn arrival dates were corrected based on actual availability, enabling a fairer comparison between the model and the company's plans.

As shown in Fig. 2, the yarn corrected model achieved 90.8% (Dd-2) and 70.8% (Dd-8) on-time completion, improving by 1.9 and 3.7% points over the company's plans.

These results demonstrate that scheduling performance is highly sensitive to input data reliability. However, when yarn genuinely arrives late, delivery performance cannot be recovered through scheduling alone, highlighting a limitation beyond the model's scope.

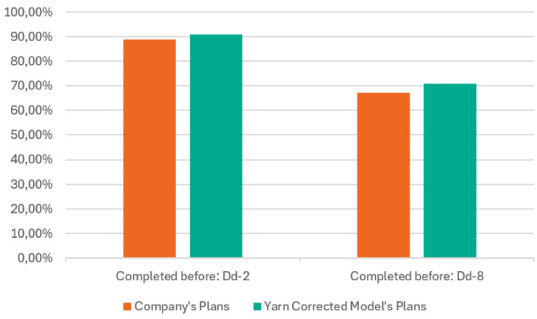

Fig. 2. Comparison of average tardiness (Dd-2 and Dd-8) between the company's plans and the yarn corrected model's plans

Finally, the makespan across different typologies was evaluated using the yarn-corrected model, as shown in Fig. 3.



Fig. 3. Makespan comparison between the company's plans and the yarn corrected model's plans

The results show a reduction in makespan for most typologies compared with the company's plans, with the largest improvements observed in Typologies 1, 4, and 5, while Typologies 2 and 3 show comparable results. This indicates that the model is particularly effective for more time-intensive typologies, contributing to improved overall efficiency.

The yarn-corrected scenarios were used solely for evaluation purposes and are not directly implementable, as they rely on manually adjusted data. This reinforces the need for improved data integration and reliability to fully exploit the model's potential in practice.

Regarding implementation, the model has been integrated into the company's planning workflow. However, its current use is limited by the lack of structured product typology data, requiring manual assignment of compatible robots and preventing full automation. Input data are provided through an Excel interface connected to the company's database, after which the optimization application generates production plans for review and integration into the planning system.

6 Conclusion and Future Work

This work developed an optimization-based scheduling model for the robot-tufted production process in a luxury carpet manufacturing company. The model generates sequenced production plans while considering key operational constraints such as robot compatibility, yarn availability, setup times, stoppages, and delivery deadlines. The results show that, when reliable yarn availability data are available, the model can improve scheduling performance compared with the company's current manual planning approach, achieving higher on-time completion rates and reduced makespan in several production typologies.

Beyond its practical contribution, this work also addresses a research gap in the literature, as optimization-based scheduling for robot-tufted carpet production has not been extensively explored.

Future work should focus on advancing toward a fully automated scheduling process by integrating product typology directly from the design stage and improving the reliability of yarn availability data. Additionally, further research could explore improved strategies for handling special carpets, such as descending and spliced products, to further enhance scheduling efficiency.

Acknowledgments. This research was developed in collaboration with LTPlabs, a consulting company where the work was carried out and was supported by FCT – Fundação para a Ciência e Tecnologia under the R&D Unit Project Scope UID/00319/2025 - Centro ALGORITMI (ALGORITMI/UM), 10.54499/UID/00319/2025

References

1. Lorente-Leyva, L.L., Alemany, M.M.E., Peluffo-Ordóñez, D.H.: A conceptual framework for the operations planning of the textile supply chains: insights for sustainable and smart planning in uncertain and dynamic contexts. Comput. Ind. Eng. **187**, 109824 (2024). https://doi.org/10.1016/j.cie.2023.109824
2. Ray, S.: A quick review of machine learning algorithms. In: 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), pp. 35–39 (2019). https://doi.org/10.1109/COMITCon.2019.8862451
3. Pinedo, M.L.: Scheduling: Theory, Algorithms, and Systems, 6th edn. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-05921-6
4. Vieira, M., Pinto-Varela, T., Barbosa-Póvoa, A.P.: A model-based decision support framework for the optimisation of production planning in the biopharmaceutical industry. Comput. Ind. Eng. **129**, 354–367 (2019). https://doi.org/10.1016/j.cie.2019.01.045

5. Berthier, A., Yalaoui, A., Chehade, H., Yalaoui, F., Amodeo, L., Bouillot, C.: Unrelated parallel machines scheduling with dependent setup times in textile industry. Comput. Ind. Eng. **174**, 108736 (2022). https://doi.org/10.1016/j.cie.2022.108736
6. Nie, Y., Biegler, L.T., Wassick, J.M.: Integrated scheduling and dynamic optimization of batch processes using state equipment networks. AIChE J. **58**(11), 3416–3432 (2012). https://doi.org/10.1002/aic.13738
7. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey. In: Hammer, P.L., Johnson, E.L., Korte, B.H. (eds.) Discrete Optimization II. Annals of Discrete Mathematics, vol. 5, pp. 287–326. Elsevier (1979). https://doi.org/10.1016/S0167-5060(08)70356-X
8. Allahverdi, A., Ng, C.T., Cheng, T.C.E., Kovalyov, M.Y.: A survey of scheduling problems with setup times or costs. Eur. J. Oper. Res. **187**(3), 985–1032 (2008). https://doi.org/10.1016/j.ejor.2006.06.060
9. Kaban, A.K., Othman, Z., Rohmah, D.S.: Comparison of dispatching rules in job-shop scheduling problem using simulation: a case study. Int. J. Simul. Model. **11**(3), 129–140 (2012). https://doi.org/10.2507/IJSIMM11(3)2.201
10. Floudas, C.A., Lin, X.: Mixed integer linear programming in process scheduling: modeling, algorithms, and applications. Ann. Oper. Res. **139**(1), 131–162 (2005). https://doi.org/10.1007/s10479-005-3446-x
11. Wilson, J.M.: Alternative formulations of a flow-shop scheduling problem. J. Oper. Res. Soc. **40**(4), 395–399 (1989). https://doi.org/10.2307/2583256
12. Manne, A.S.: On the job-shop scheduling problem. Oper. Res. **8**(2), 219–223 (1960). https://doi.org/10.1287/opre.8.2.219
13. Pan, C.H.: A study of integer programming formulations for scheduling problems. Int. J. Syst. Sci. **28**(1), 33–41 (1997). https://doi.org/10.1080/00207729708929360
14. Frühwirth, T., Abdennadher, S.: Essentials of Constraint Programming. Springer, Berlin Heidelberg (2003). https://doi.org/10.1007/978-3-662-05138-2
15. Glover, F.: Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. **13**(5), 533–549 (1986). https://doi.org/10.1016/0305-0548(86)90048-1
16. Fonseca, G.H.G., Figueiroa, G.B., Toffolo, T.A.M.: A fix-and-optimize heuristic for the unrelated parallel machine scheduling problem. Comput. Oper. Res. **163**, 106504 (2024). https://doi.org/10.1016/j.cor.2023.106504
17. Ploskas, N.: Parameter tuning of linear programming solvers. In: 2022 7th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM), pp. 1–6 (2022). https://doi.org/10.1109/SEEDA-CECNSM57760.2022.9933002
18. Rossi, A., Tiacci, L., Simonetti, M.: A mixed integer programming model to optimize production planning in the luxury textile industry. Procedia Comput. Sci. **253**, 1175–1184 (2025). https://doi.org/10.1016/j.procs.2025.01.179
19. Avgerinos, I., Mourtos, I., Vatikiotis, S., Zois, G.: Weighted tardiness minimization for unrelated machines with sequence-dependent and resource-constrained setups. arXiv preprint arXiv:2307.06671 (2023)
20. Williams, R., Pitts, S., Kamery, R.: Production scheduling problems for a totally integrated carpet manufacturer: a preliminary investigation (2004)