Journal of Industrial and Management Optimization

JIMO, 22(10): 4627–4675
DOI: 10.3934/jimo.2026161
Received: 09 July 2026
Revised: 10 August 2026
Accepted: 21 August 2026
Published: 01 September 2026

https://www.aimspress.com/journal/jimo

Research article

Optimization model for injection molding production scheduling in kitchen-appliance manufacturing: A real-world case study

M. Sankar[1] and M. Ramakrishnan[2,*]

[1] Vidyaa Vikas College of Engineering and Technology, Tiruchengode, Namakkal District, Tamil Nadu, India
[2] K.S. Rangasamy College of Technology, Tiruchengode, Namakkal District, Tamil Nadu, India

* **Correspondence:** Email: ramakrishnan@ksrct.ac.in.

Abstract: Production scheduling in injection molding is challenging when production capacity depends not only on machines, but also on secondary shared tooling such as molds. In this paper, we present a real-world case study of a mixed-integer programming model developed for a kitchen-appliance manufacturing company to generate feasible and cost-efficient injection molding production schedules. The proposed formulation integrated machine assignment, mold allocation, sequence-dependent setup decisions, production quantities, inventory balance, subcontracting, planned stoppages, mold maintenance, shift calendars, holidays, and grouped customer-level demand fulfillment within a single optimization framework. The novelty of the study lies in representing the coupled machine–tooling–demand structure of an industrial molding environment in a unified model, rather than treating capacity planning, mold usage, sequencing, and demand fulfillment as separate planning problems. Although motivated by injection molding in kitchen-appliance manufacturing, the modeling approach is broadly applicable to industries where production depends on secondary shared tooling, including automotive components, consumer durables, electrical products, packaging, medical devices, metal forming, die casting, and other high-mix manufacturing environments. To the best of our knowledge, this is the first unified mixed-integer programming formulation for injection-molding production scheduling that integrates shared molds, sequence-dependent setups, operational disruptions, inventory, subcontracting, and grouped customer demand.

Keywords: production scheduling; optimization model; mixed-integer programming; sequence-dependent setups; integrated planning models; injection molding
Mathematics Subject Classification: 90C11, 90B35, 90B30, 90C90, 90C27

1. Introduction

Injection molding is one of the most important processing technologies within the global plastics industry. Plastics production has expanded to a very large industrial scale over the last several decades, and plastic products now serve as essential inputs in automotive, packaging, healthcare, electrical and electronics, household, and construction applications [1–3]. Within this broad industrial ecosystem, injection molding occupies a central position because it enables high-volume, repeatable production of geometrically complex parts with tight dimensional tolerances and good surface quality [4,5]. The process is especially attractive when manufacturers require rapid cycle times, low unit cost at scale, and robust compatibility with automated production systems [4–7].

The industrial significance of injection molding is reflected in the diversity of its applications. In automotive manufacturing, molded polymer parts are used extensively in interior modules, trims, housings, clips, ducts, and lightweight functional components that contribute to mass reduction and design flexibility [2,4]. In the medical domain, injection molding supports high-volume production of disposable and precision plastic items such as syringes, diagnostic cartridges, device housings, and sterile consumables [6]. Packaging and consumer electronics similarly rely on the process for large-scale, repeatable manufacture of standardized yet intricate products [2,5]. As a result, even moderate improvements in production planning and schedule quality can translate into substantial gains in machine utilization, mold utilization, delivery reliability, and operating cost.

Despite the apparent repetitiveness of the molding cycle, production planning in injection molding plants is structurally complex. Feasibility is governed not by machine capacity alone, but by the simultaneous availability of machines, molds, compatible material conditions, setup states, and production time [4,7]. In many practical environments, a mold can be mounted only on one machine at a time and is compatible with only a subset of presses. Molds are therefore not merely passive tools; they are scarce and capacity-constraining resources that must be planned jointly with machines [8,9]. This dual-resource dependency differentiates injection molding from simpler parallel-machine settings and substantially increases scheduling complexity.

A second major source of difficulty is the prevalence of sequence-dependent setup operations. Changing from one product to another typically involves mold change, material purging, temperature stabilization, process resetting, and quality validation. These activities consume non-productive time and often depend strongly on the production sequence [10,11]. In high-mix environments, setup losses can absorb a significant share of effective capacity, so models that ignore sequencing effects may underestimate resource requirements and generate schedules that are infeasible in practice [12,13].

Production planning in injection molding is also shaped by the classical tension between setup efficiency and inventory responsiveness. Longer campaigns reduce the number of changeovers but raise inventory carrying and obsolescence risk. Shorter campaigns improve responsiveness to demand variation but increase setup frequency and capacity loss [9,10]. This trade-off naturally connects injection molding scheduling to the broader literature on capacitated lot sizing and integrated lot-sizing–scheduling models [14–16].

For these reasons, injection molding scheduling should be treated as an integrated operations research problem rather than as a simple dispatching exercise. Practical solution approaches must coordinate machine assignment, mold allocation, setup sequencing, production quantities, and time-phased inventory decisions within a single planning framework.

2. Literature review

The most relevant theoretical foundation for injection molding production planning comes from the literature on lot sizing and scheduling. Early production planning research established the importance of optimizing production quantities across time rather than making isolated period-by-period decisions. Wagner and Whitin [17] provided the classical dynamic lot-sizing framework, while Manne [18] introduced integer programming ideas that helped shape later scheduling formulations. These studies did not model injection molding directly, but they established the optimization logic required for integrated planning.

Capacitated lot sizing extended this foundation by recognizing that production decisions must respect finite resource availability. Karimi et al. [14] surveyed the capacitated lot-sizing problem and documented the major formulation families and algorithmic issues. Drexl and Kimms [15] then showed that lot sizing and scheduling are most realistically treated as a connected problem class rather than as separate decisions. Their classification remains highly relevant for injection molding because sequence-dependent setups and resource coupling make decomposition conceptually attractive but often operationally weak.

A major stream of the literature has focused on setup modeling. Allahverdi et al. [10] reviewed scheduling research involving setup considerations and demonstrated the breadth of environments in which setup times and setup costs materially affect scheduling performance. Their later survey [11] broadened the analysis and became one of the standard references for sequence-dependent setup scheduling. Allahverdi [12] subsequently updated this literature and confirmed that setup-aware modeling remains essential in practical scheduling applications. This body of work is directly relevant to injection molding, where mold changeovers and related start-up activities are often costly and sequence dependent.

Integrated lot-sizing and scheduling formulations emerged precisely because aggregate lot-sizing models were insufficient when sequencing effects mattered. Meyr [16] proposed simultaneous lot-sizing and scheduling formulations that explicitly connect production quantities with setup decisions. Pochet and Wolsey [19] then provided a rigorous mixed-integer programming treatment of production planning, including strong formulations that have influenced a large share of later work. These contributions are central for injection molding because they demonstrate how binary setup variables, material balance equations, and capacity constraints can be embedded in a common optimization framework.

Resource interactions add another layer of difficulty. The scheduling literature summarized by Pinedo [20] and by Lawler et al. [21] further clarifies why such environments are difficult: Once parallel resources, precedence logic, and setup effects are combined, even simplified variants quickly become NP-hard. Framinan et al. [22] provide a review and classification of heuristic methods for permutation flow-shop scheduling with makespan minimization, offering a useful background for heuristic scheduling approaches.

Because exact optimization becomes difficult on large instances, heuristic and metaheuristic methods have played an important role in the broader scheduling and lot-sizing literature. Holland's genetic algorithm framework [23], Glover's tabu search [24], and Kirkpatrick et al.'s simulated annealing method [25] motivated many later applications to integrated planning and sequencing problems. Jans and Degraeve [26] reviewed metaheuristics for dynamic lot sizing and showed that these methods can be effective when exact methods face scale limitations. For rolling and uncertain environments, Sox and Muckstadt [27] demonstrated how demand uncertainty alters lot-sizing decisions, while Tempelmeier [28] addressed inventory and capacity planning under dynamic conditions. These ideas matter for injection molding because planners frequently operate in rolling-horizon settings with volatile product mixes and recurring rescheduling. Copil et al. [29] present a detailed classification and review of simultaneous lot-sizing and scheduling models, making this work highly relevant to integrated production planning formulations.

Another important research direction concerns hybrid and decomposition-based optimization. Hooker [30] showed how mixed-integer programming and constraint programming can be combined in planning and scheduling contexts, while Kogan [31] illustrated how additional resource restrictions alter feasible schedules even in parallel-machine environments. These works are useful for injection molding because exact integrated models may require decomposition, hybridization, or tailored reformulations to achieve computational practicality.

Within the lot-sizing–scheduling literature, a substantial body of researchers has addressed sequence-dependent setups in greater detail. Fleischmann [32] studied the discrete lot-sizing and scheduling problem with sequence-dependent setup costs, clarifying why sequencing cannot be ignored when setup structure is material. Fleischmann and Meyr [33] further characterized the general lotsizing and scheduling problem and helped standardize later modeling approaches. Haase and Kimms [34] examined sequence-dependent setup costs and times together with rescheduling opportunities, which is especially relevant for plants that repeatedly revise plans in response to operational changes.

Several studies then strengthened the connection to multi-product production environments. Gupta and Magnusson [35] addressed the capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times. Almada-Lobo et al. [36] examined single-machine multi-product capacitated lot sizing with sequence-dependent setups, while Clark and Clark [37] investigated rolling-horizon lot sizing under the same type of setup dependence. These contributions are highly relevant to injection molding because they formalize the interaction between campaign sizing, setup sequencing, and periodic replanning.

Setup carryover has also been recognized as an important modeling issue. Gopalakrishnan et al. [38] proposed a framework for setup carryover in the capacitated lot-sizing problem, and Gopalakrishnan [39] later refined that framework. These studies matter in injection molding whenever the ending setup state of one planning period affects the cost and feasibility of the next period's production sequence.

Finally, multi-level and more integrated production settings have been studied in ways that are also relevant to injection molding. Fandel and Stammen-Hegene [40] addressed simultaneous lot sizing and scheduling for multi-product, multi-level production. Anwar and Nagi [41] considered integrated lot sizing and scheduling for just-in-time production of complex assemblies with finite setups. Kogan [42] studied scheduling under a common due date with a single constrained resource and precedence constraints, showing how additional resource restrictions can significantly affect feasible scheduling decisions. Stadtler [9] showed how setup times and multiple constrained resources can be handled together in a multilevel planning model. Taken together, these studies make two points clear. First, realistic

production planning requires simultaneous treatment of quantities, sequencing, and constrained resources. Second, despite the maturity of the general literature, there remains strong motivation to develop formulations tailored to injection molding, where mold availability, machine–mold compatibility, and sequence-dependent changeovers jointly determine feasibility.

Studies provide useful points of comparison for positioning this formulation. Dastidar and Nagi [43] addressed injection-molding scheduling with parallel workcenters, multiple capacitated resources, sequence-dependent setup times and costs, inventory, and backlogging, and proposed a decomposition approach for industrial-scale instances. Winklehner and Hauder [44] subsequently studied a real-world flexible job-shop scheduling problem in an automotive manufacturer producing injection-molded parts, incorporating alternative resources, release dates, deadlines, and sequence-dependent setup times, and demonstrated the effectiveness of constraint programming for that setting. Moreover, Troncoso et al. [45] developed the product-mold-machine manufacturing problem, explicitly recognizing limited molds, machine-mold compatibility, changeover times, and the coupling among products, molds, and machines. Their manufacturing structure differs from the injection-molding setting because a machine may process multiple molds concurrently and the principal planning objective is makespan-oriented. These studies demonstrate the importance of secondary-resource and setup-aware scheduling while addressing different subsets of the operational decisions integrated in the present formulation.

Research also reflects a broader movement toward integrated and industrially validated production decision models. Bok et al. [46] formulated a bi-objective mixed-integer model for high-mix, low-volume production on non-identical parallel machines, jointly considering production cost, due dates, energy sources, and carbon emissions and validating the approach with industrial data. Kim et al. [47] extended this direction to multi-stage production with unrelated parallel machines under carbon regulation and developed a hybrid ALNS-MILP approach to improve scalability. Jo et al. [48], although concerned with industrial steam procurement rather than production sequencing, integrated forecasting, supplier evaluation, order allocation, cost, and carbon-emission decisions, illustrating the increasing emphasis on unified decision-support frameworks that connect operations with external sourcing decisions. Injection-molding-specific research has also continued: Capponi et al. [49] considered a real-life batching and sequencing problem for automotive injection molding and developed ant-colony optimization approaches for industrial-size instances. At a broader smart-manufacturing level, Akbari et al. [50] proposed a data-driven dynamic job-shop framework integrating real-time order acceptance, machine deterioration, maintenance-related availability, energy consumption, and raw-material inventory, with validation in an industrial case. Among the representative studies reviewed above, none simultaneously integrates eligible machines, shared physical molds, sequence-dependent setups, planned stoppages and holidays, mold maintenance, inventory and backlog, subcontracting, and grouped customer-level demand completion within a single scheduling formulation. This combination therefore provides the principal structural distinction of the present model.

From a smart-manufacturing perspective, recent data-driven scheduling research also provides a pathway from deterministic optimization toward dynamic shop-floor decision support. Akbari et al. [50], for example, integrate real-time order acceptance, machine deterioration, maintenance-related availability, energy consumption, and inventory information within a dynamic scheduling framework. This perspective is relevant to the present model because several of its inputs, including machine and mold availability, cycle and setup times, inventory states, demand, completed production, and planned disruptions, can be refreshed as new shop-floor information becomes available. Thus, although we evaluate a deterministic formulation, the model can serve as the optimization core of a rolling-horizon

decision architecture in which updated information from a Manufacturing Execution System (MES) is periodically supplied to the scheduler and the remaining horizon is re-optimized. Such an architecture also provides a natural interface for future data-driven or predictive models that estimate changing processing times, setup durations, equipment availability, or demand before each rescheduling cycle.

Figure 1 shows a capability-based comparison between representative related studies and this work. The comparison is intended to identify structural coverage rather than to rank the individual studies. A green circle indicates that a capability constitutes a defining component of the corresponding study, a yellow circle denotes partial or application-specific treatment, and a red cross indicates that the capability is not a defining feature of that formulation. Dastidar and Nagi [43] provided an important injection-molding benchmark through their treatment of sequence-dependent setups, multiple resource constraints, inventory, and backlog; Winklehner and Hauder [44] emphasized flexible machine assignment and sequence-dependent setup scheduling in an industrial injection-molding environment; and Troncoso et al. [45] provided an explicit representation of product-mold-machine coupling. Bok et al. [46] and Kim et al. [47] broadened the comparison toward heterogeneous-machine industrial scheduling under sustainability considerations, while Jo et al. [48] illustrated the integration of external sourcing and allocation decisions. The comparison highlights that the contribution of this study does not arise from any one modeling element in isolation, but from their joint representation within a machine-tooling-demand scheduling framework that includes shared physical molds, sequence-dependent setups, operational disruptions, inventory and backlog, subcontracting, and grouped customer-level demand fulfillment.

Figure 1. Capability-based comparison of representative related studies and the present formulation.

3. Mathematical formulation

In this section, we describe the MIP model for injection molding production scheduling. We begin with the problem statement, followed by the sets, parameters and decision variables used in the study. Based on the above, a complete MIP model is proposed and discussed.

3.1. Problem statement

Injection molding operations involve tightly coupled decisions across machines, molds, products, demand, setup states, and operating calendars. In practical plants, production feasibility depends not only on machine capacity but also on the availability of compatible molds, the sequence in which products are scheduled, planned maintenance, stoppages, inventory requirements, subcontracting options, and customer-level fulfillment commitments. The optimization model developed in this study captures these interdependencies within a unified mixed-integer programming framework to generate feasible and cost-efficient schedules over a finite planning horizon.

The objective of the model is to minimize the total cost of meeting item-level and grouped demand while satisfying the physical, temporal, and logical constraints of injection molding production. The cost structure includes in-house production, inventory holding, subcontracting, penalties for undesired outsourcing, backlog, and delayed completion of grouped customer demand. This enables the model to balance production efficiency, inventory stability, outsourcing decisions, and customer service performance within a single planning criterion.

The planning horizon is divided into periods and further into subperiods or machine slots. This slot-based structure enables detailed sequencing decisions to be represented at the machine level while retaining period-level constraints such as shifts, holidays, and rolling-horizon commitments. Each machine slot is assigned exactly one item state, reflecting the physical limitation that an injection molding machine can process only one mold–item combination at a time.

A central feature of the formulation is the explicit treatment of sequence-dependent setup times. When a machine switches from one item to another, the corresponding setup duration is determined by the item-to-item transition. The setup time is computed at the slot level and reduces the effective processing time available for production. Consequently, the model captures the operational trade-off between frequent changeovers, campaign length, and productive capacity utilization.

Mold availability and compatibility are also modeled explicitly. Each item may require one of several eligible molds, and each mold may be compatible with only a subset of machines. The formulation prevents the same physical mold from being used on more than one machine in the same subperiod and links production quantities directly to mold assignment. For items with multiple eligible molds, item-level and mold-level production variables are connected to preserve physical feasibility and avoid aggregation errors.

Capacity feasibility is enforced through time-window, continuity, shift, holiday, and stoppage constraints. Slot start and end times are restricted to their assigned periods, and consecutive slots are linked to prevent infeasible overlap or idle gaps except where allowed by shift transitions. Shift-related constraints ensure compliance with one-, two-, or three-shift operating patterns. Holidays force production and setup times to zero while preserving the item-selection state for continuity. Planned machine stoppages are embedded into the slot sequence by assigning a zero-production slot to each stoppage interval; during such intervals, both production and setup activity are prohibited.

Inventory and demand fulfillment are represented through balance equations that account for initial inventory, in-house production, subcontracting, inventory carryover, and backlog. The model

supports rolling-horizon use by incorporating commitment boundaries and inventory balance logic that maintain consistency across successive optimization runs. Minimum ending inventory requirements may also be imposed to improve plan stability and avoid short-term depletion of critical items.

Although the formulation is deterministic within each optimization run, it is designed to support repeated rolling-horizon execution. At each rescheduling epoch, information received from an MES or other shop-floor information system can be used to update demand, inventory, completed production, machine and mold availability, planned or newly identified stoppages, and, where appropriate, cycle and setup times. The remaining planning horizon can then be re-optimized using the updated state. Decisions already executed or operationally committed are retained through the commitment-boundary logic, while inventory and machine-state information from the preceding horizon provide continuity between successive schedules. Consequently, dynamic rescheduling does not require a different mathematical formulation; it can be implemented by repeatedly solving the present model with refreshed parameters and updated initial states. Real-time MES connectivity, event-triggered rescheduling, and predictive updating of uncertain processing and setup parameters are therefore natural extensions of the present implementation rather than assumptions made in the computational experiments reported here.

Subcontracting is included as a flexible but controlled demand-fulfillment option. The model respects already committed outsourced quantities while allowing additional outsourcing when internal capacity is insufficient or economically unfavorable. Penalty terms discourage subcontracting of items preferred for in-house manufacture, thereby representing strategic sourcing, quality, or customer requirements without creating unnecessary infeasibility.

The formulation further extends item-level planning to grouped customer demand through header-level structures. Items may belong to one or more headers, and each header demand may involve multiple items across periods. Additional variables allocate inventory, production, and subcontracted quantities to specific header demands, while completion variables identify the period in which each grouped demand is satisfied. This enables the model to penalize late completion and treat customer-level requirements coherently rather than as independent item demands.

Maintenance constraints restrict mold assignment and production when molds are unavailable, and optional logic can propagate maintenance effects to header-level fulfillment decisions. Overall, the proposed model provides an integrated representation of industrial injection molding scheduling by combining machine-slot sequencing, shared mold allocation, setup-dependent capacity, inventory dynamics, subcontracting, stoppages, maintenance, and grouped demand fulfillment in one formulation. The formulation is therefore relevant not only to injection molding but also to other manufacturing systems where production depends on secondary shared tooling and time-phased demand. This makes the model suitable for complex manufacturing environments in which scarce tooling, operational disruptions, and competing service and cost objectives must be planned simultaneously.

3.2. Sets

- M: Set of injection molding machines available for production.
- I: Set of items to be produced to satisfy demand.
- K: Set of molds used in the injection molding process.
- EIM_m: Set of items eligible to be produced on machine $m,\ \forall m \in M$.
- EIK_k: Set of items eligible to be produced using mold $k,\ \forall k \in K$.
- EMK_k: Set of machines eligible to use mold $k,\ \forall k \in K$.

- EMI_i: Set of machines eligible to produce item i, $\forall i \in I$.
- EKI_i: Set of molds eligible to produce item i, $\forall i \in I$.
- $EKM_m = \{k \in K \mid m \in EMK_k\}$: Set of molds eligible to be used on machine m, $\forall m \in M$.
- T: Set of periods in the planning horizon.
- S_t: Set of subperiods (time slots) in period t, $\forall t \in T$.
- $S = \bigcup_{t \in T} S_t$: Set of all subperiods across the planning horizon.
- $STOP_{tm}$: Set of stoppages occurring in period t on machine m, $\forall t \in T, m \in M$.
- HC: Set of header codes representing grouped customer demand.
- HI_i: Set of headers associated with item i, $\forall i \in I$.
- IH_h: Set of items associated with header h, $\forall h \in HC$.

3.3. Parameters

- ϕ: Number of subperiods (time slots) in one period.
- d_{it}: Demand for item i in period t, $\forall i \in I, t \in T$.
- ct_{im}: Cycle time for producing item i on machine m, $\forall i \in I, m \in EMI_i$.
- c_k: Number of cavities in mold k, $\forall k \in K$.
- sc_i: Subcontracting cost per unit of item i, $\forall i \in I$.
- ihc_i: In-house production cost per unit of item i, $\forall i \in I$.
- icc_i: Inventory carrying cost per unit of item i, $\forall i \in I$.
- $I0_i$: Initial inventory of item i at time $t = 0$, $\forall i \in I$.
- $source_i$: Desired sourcing mode (In-house or outsourced) for item i, $\forall i \in I$.
- $initialItem_m$: Item running on machine m at time $t = 0$, $\forall m \in M$.
- $setuptime_{ijm}$: Setup time in hours required to switch from item i to item j on machine m, $\forall m \in M, i, j \in EIM_m$.
- cap: Time capacity of one period.
- ss_r: Start time of stoppage r, $\forall t \in T, m \in M, r \in STOP_{tm}$.
- se_r: End time of stoppage r, $\forall t \in T, m \in M, r \in STOP_{tm}$.
- v_t: Binary indicator equal to 1 if period t is a holiday, $\forall t \in T$.
- mm_{kt}: Binary indicator equal to 1 if mold k is under maintenance in period t, $\forall k \in K, t \in T$.
- f_m: The last scheduled production time of machine m in the previous planning horizon, $\forall m \in M$.
- AO_{it}: Quantity of item i already subcontracted in period t, $\forall i \in I, t \in T$.
- IE_i: Required inventory of item i at the end of planning horizon, $\forall i \in I$.
- sh_t: Number of shifts planned in period t, $\forall t \in T$.
- $startFS$: Start time of first shift.
- $endFS$: End time of first shift.
- $endSS$: End time of second shift.
- IHD_{hti}: Demand of item i toward header h in period t, $\forall h \in HC, t \in T, i \in IH_h$.
- mmo_k: Indicator defining whether related headers must stop when mold k is under maintenance, $\forall k \in K$.
- T_f: Final period in the planning horizon.
- $BigM$: A sufficiently large positive constant used in Big-M logical implication and activation constraints.
- $p1$: Penalty multiplier applied when an item whose desired sourcing mode is in-house is subcontracted.
- $p2$: Penalty coefficient applied per period of delay in completing grouped header-level demand.

- $p3_i$: Backlog penalty cost per unit of item i per period, $\forall i \in I$.

3.4. Decision variables

- ts_{sm}: Start time of subperiod s on machine m, $\forall s \in S, m \in M$.
- te_{sm}: End time of subperiod s on machine m, $\forall s \in S, m \in M$.
- n_{sm}: Length of subperiod s on machine m *in hours*, $\forall s \in S, m \in M$.
- p_{ism}: Quantity of item i produced in subperiod s on machine m, $\forall s \in S, m \in M, i \in EIM_m$.
- y_{ism}: Binary variable equal to 1 if item i is produced in subperiod s on machine m, $\forall s \in S, m \in M, i \in EIM_m$.
- z_{ijsm}: Binary variable equal to 1 if changeover from item i to item j occurs in subperiod s on machine m, $\forall s \in S, m \in M, i, j \in EIM_m$.
- pk_{ismk}: Quantity of item i produced using mold k in subperiod s on machine m, $\forall k \in K, s \in S, m \in EMK_k, i \in EIK_k \cap EIM_m$.
- b_{ksm}: Binary variable equal to 1 if mold k is used on machine m in subperiod s, $\forall k \in K, s \in S, m \in EMK_k$.
- st_{sm}: Setup time incurred in subperiod s on machine m, $\forall s \in S, m \in M$.
- I_{it}: Inventory of item i at end of period t, $\forall i \in I, t \in T$.
- B_{it}: Backlog of item i at end of period t, $\forall i \in I, t \in T$.
- O_{it}: Subcontracted quantity of item i in period t, $\forall i \in I, t \in T$.
- ls_{rsm}: Binary indicator that slot s ends before stoppage r, $\forall t \in T, s \in S_t, m \in M, r \in STOP_{tm}$.
- rs_{rsm}: Binary indicator that slot s starts after stoppage r, $\forall t \in T, s \in S_t, m \in M, r \in STOP_{tm}$.
- bs_{rsm}: Binary indicator that slot overlaps with stoppage r, $\forall t \in T, s \in S_t, m \in M, r \in STOP_{tm}$.
- ia_{ith}: Inventory of item i allocated to header h demand in period t, $\forall i \in I, t \in T, h \in HI_i$.
- $pa_{it\tau h}$: Quantity of item i produced in period τ to satisfy header h demand in period t, $\forall i \in I, t \in T, \tau \in T, h \in HI_i$.
- oa_{ith}: Subcontracted quantity of item i allocated to header h demand in period t, $\forall i \in I, t \in T, h \in HI_i$.
- $hf_{ht\tau}$: Binary variable equal to 1 if header h demand in period t is completed in period τ, $\forall h \in HC, t \in T, \tau \in T \mid \tau \geq t$.

3.5. Objective function

Minimize

$$Z = \sum_{i \in I} \sum_{t \in T} s\, c_i O_{it} + \sum_{i \in I} \sum_{t \in T} i\, cc_i I_{it} + \sum_{s \in S} \sum_{m \in M} \sum_{i \in EIM_m} i\, hc_i p_{ism}$$
$$+ \sum_{i \in I \mid source_i = \text{In-house}} \sum_{t \in T} p\, 1\, sc_i O_{it} + \sum_{h \in HC} \sum_{t \in T} \sum_{\tau \in T \mid \tau \geq t} p\, 2(\tau - t) hf_{ht\tau} + \sum_{i \in I} \sum_{t \in T} p\, 3_i B_{it}$$

The objective function minimizes the total cost of operating the injection molding system while satisfying all production and demand requirements. It includes the cost of subcontracting items, which captures the expense of outsourcing production when in-house capacity is insufficient or strategically undesirable. Inventory carrying costs are incorporated to discourage excessive stock accumulation and to promote timely production aligned with demand. The function also accounts for in-house production

costs, directly linking machine-level production quantities to operational expenditure. Additional penalty terms are applied when items that are designated for in-house production are instead subcontracted, reflecting quality or sourcing preferences. Furthermore, the objective penalizes delayed completion of grouped (header-level) demand, with the penalty increasing in proportion to the delay. This encourages earlier fulfillment of customer orders and improves service levels. The objective also includes a backlog penalty term to discourage unmet demand at the end of the planning horizon or across periods, depending on the selected backlog-penalty structure. By combining these cost components, the objective function balances production efficiency, inventory stability, outsourcing decisions, and customer satisfaction within a single optimization criterion.

3.6. Constraints

3.6.1. Periods and slots

$$(t-1)\cdot cap \le ts_{sm} \le t\cdot cap, \quad \forall t \in T, s \in S_t, m \in M \tag{1}$$

This constraint ensures that the start time of each subperiod lies within the boundaries of its corresponding planning period. It prevents slots from starting before the period begins or after the period ends.

$$(t-1)\cdot cap \le te_{sm} \le t\cdot cap, \quad \forall t \in T, s \in S_t, m \in M \tag{2}$$

This constraint restricts the end time of each subperiod to remain within the same planning period. Together with (1), it guarantees that every slot is fully contained inside its assigned period.

$$ts_{sm} = te_{s-1,m}, \quad \forall t \in T, s \in S_t, m \in M \mid s \neq (t-1)\cdot \varphi + 1 \tag{3}$$

This constraint enforces exact temporal continuity between consecutive slots on the same machine within a period. The start time of a slot is equal to the end time of the previous slot, so arbitrary idle gaps between consecutive slots are not allowed. Planned stoppages are represented separately through zero-production stoppage slots whose start and end times coincide with the corresponding stoppage interval.

$$n_{sm} = te_{sm} - ts_{sm}, \quad \forall s \in S, m \in M \tag{4}$$

This defines the duration of each subperiod as the difference between its end and start times. The slot length is later used to compute the available processing time.

$$ts_{1,m} = f_m, \quad \forall m \in M \tag{5}$$

This constraint links the first slot of the planning horizon to the finished production time of the previous horizon. It ensures temporal continuity across rolling-horizon optimization runs.

$$te_{sm} \le (t-1)\cdot cap + \ endFS, \quad \forall t \in T, s \in S_t, m \in M \mid sh_t = 1 \tag{6}$$

When only one shift is planned in a period, this constraint ensures that all production finishes within the first shift. No slot is allowed to extend beyond the shift boundary.

$$te_{sm} \le (t-1)\cdot cap + \ endSS, \quad \forall t \in T, s \in S_t, m \in M \mid sh_t = 2 \tag{7}$$

For periods with two shifts, this constraint allows production to extend until the end of the second shift. It enforces compliance with shift-level working-hour limits.

$$ts_{sm} \geq (t-1) \cdot cap + \ startFS, \\ \forall t \in T \backslash \{1\}, s \in S_t, m \in M \mid sh_{t-1} = 1, s = (t-1) \cdot \varphi + 1 \tag{8}$$

This constraint ensures that production in the next period does not start before the first shift begins. It applies when the previous period had only one shift.

$$ts_{sm} \geq (t-1) \cdot cap + \ startFS, \\ \forall t \in T \backslash \{1\}, s \in S_t, m \in M \mid sh_{t-1} = 2, s = (t-1) \cdot \varphi + 1 \tag{9}$$

This constraint enforces the same shift-start condition when the previous period had two shifts. It prevents illegal carryover of production into non-working hours.

$$ts_{sm} = te_{s-1,m}, \quad \forall t \in T \backslash \{1\}, s \in S_t, m \in M \mid sh_{t-1} = 3, s = (t-1) \cdot \varphi + 1 \tag{10}$$

This constraint enforces continuity across periods when production spans three shifts. The first slot of the new period starts immediately after the last slot of the previous period.

3.6.2. Injection molding

$$\sum_{i \in EIM_m} y_{ism} = 1, \forall s \in S, m \in M \tag{11}$$

This constraint ensures that exactly one item state or machine configuration is selected in each slot on each machine. It enforces single-item production per machine per subperiod.

$$y_{initialItem_m,0,m} = 1, \quad \forall m \in M \tag{12}$$

$$y_{i,0,m} = 0, \quad \forall m \in M, i \in EIM_m \mid i \neq initialItem_m \tag{13}$$

This initializes the production state of each machine at the start of the horizon. It specifies the item that is already running on the machine.

$$\sum_{j \in EIM_m} z_{ijsm} = y_{i,s-1,m}, \forall s \in S, m \in M, i \in EIM_m \tag{14}$$

This constraint links changeovers to the item produced in the previous slot. A changeover can occur only from the item that was previously running.

$$\sum_{i \in EIM_m} z_{ijsm} = y_{jsm}, \forall s \in S, m \in M, j \in EIM_m \tag{15}$$

This constraint ensures that a changeover leads to the item produced in the current slot. Together with (13), it forms a valid item-to-item transition.

$$st_{sm} = \sum_{i \in EIM_m} \sum_{j \in EIM_m} setuptime_{ijm} \ z_{ijsm}, \forall s \in S, m \in M \tag{16}$$

This computes the setup time incurred in each slot based on the selected changeover. It captures sequence-dependent setup times.

$$p_{ism} \leq BigM \cdot y_{ism}, \quad \forall s \in S, m \in M, i \in EIM_m \tag{17}$$

This constraint allows production of item i only if it is selected for that slot. If the item is not active, its production quantity must be zero.

$$p_{ism} \leq \text{BigM} \sum_{k \in EKI_i \cap EKM_m} b_{ksm}, \forall s \in S, m \in M, i \in EIM_m \tag{18}$$

This constraint ensures that production is possible only if a compatible mold is assigned. It links item production to mold availability.

$$\sum_{m \in EMK_k} b_{ksm} \leq 1, \forall k \in K, s \in S \tag{19}$$

The model enforces that a mold can be used on at most one machine in any subperiod, preventing simultaneous usage of the same physical mold.

$$\sum_{k \in EKM_m} b_{ksm} \leq 1, \forall m \in M, s \in S \tag{20}$$

This ensures that at most one mold is assigned to a machine. It applies when multiple molds are compatible with the item.

$$pk_{ismk} \leq BigM \cdot b_{ksm}, \quad \forall k \in K, s \in S, m \in EMK_k, i \in EIK_k \cap EIM_m \tag{21}$$

This constraint links mold usage to production quantity. Production using mold k is allowed only if the mold is assigned.

$$p_{ism} = \sum_{k \in EKI_i \cap EKM_m} p\,k_{ismk}, \forall s \in S, m \in M, i \in EIM_m \tag{22}$$

This constraint ensures that total item production is supported by mold-level production. It enforces consistency between item and mold quantities.

$$pk_{ismk} \leq (n_{sm} - st_{sm}) \cdot 3600 \cdot \frac{c_k}{ct_{im}}, \quad \forall k \in K, t \in T, s \in S_t, m \in EMK_k, i \in EIK_k \cap EIM_m \tag{23}$$

This limits mold-level production by the effective processing time of the slot. It accounts for setup time, cycle time, and mold cavity.

$$n_{sm} \geq st_{sm}, \forall s \in S, m \in M \tag{24}$$

This constraint ensures that the setup time assigned to a slot does not exceed the total length of that slot. It prevents the effective processing time, given by slot length minus setup time, from becoming negative.

$$I_{i,t-1} - B_{i,t-1} + \sum_{m \in EMI_i} \sum_{s \in S_t} p_{ism} + O_{it} = d_{it} + I_{it} - B_{it}, \forall i \in I, t \in T \tag{25}$$

This constraint represents the inventory-backlog balance for each item and period. The available quantity consists of net carryover from the previous period, in-house production during the current period, and subcontracted quantity. This available quantity is balanced against current-period demand,

ending inventory, and ending backlog. The inclusion of previous-period backlog ensures that unmet demand is carried forward correctly across periods.

$$I_{i,0} = I0_i, \quad \forall i \in I \tag{26}$$

This constraint sets the initial inventory level for each item at the beginning of the planning horizon.

$$B_{i,0} = 0, \quad \forall i \in I \tag{27}$$

This constraint sets the initial backlog of each item to zero. Together, these two constraints anchor the inventory and backlog recursion used throughout the planning horizon.

$$I_{i,T_f} \geq IE_i, \quad \forall i \in I \tag{28}$$

This constraint imposes a minimum required inventory level for each item at the final period of the planning horizon. It ensures that sufficient stock is carried forward beyond the current planning window and helps avoid short-term decisions that deplete inventory excessively near the end of the horizon.

$$O_{it} \geq AO_{it}, \quad \forall i \in I, t \in T \tag{29}$$

This ensures that already subcontracted quantities are respected. The model can increase outsourcing, but cannot reduce below AO_{it}.

3.6.3. Stoppages

$$te_{sm} \leq ss_r + BigM \cdot (1 - ls_{rsm}), \quad \forall t \in T, s \in S_t, m \in M, r \in STOP_{tm} \tag{30}$$

This constraint supports the logical classification that a slot ends before a stoppage. If $ls_{rsm} = 1$, it enforces $te_{sm} \leq ss_r$; otherwise, the BigM term relaxes the bound.

$$ts_{sm} \geq se_r - BigM \cdot (1 - rs_{rsm}), \quad \forall t \in T, s \in S_t, m \in M, r \in STOP_{tm} \tag{31}$$

This constraint supports the logical classification that a slot starts after a stoppage. If $rs_{rsm} = 1$, it enforces $ts_{sm} \geq se_r$; otherwise, the BigM term relaxes the bound.

$$ts_{sm} \geq ss_r - BigM \cdot (1 - bs_{rsm}), \quad \forall t \in T, s \in S_t, m \in M, r \in STOP_{tm} \tag{32}$$

$$ts_{sm} \leq ss_r + BigM \cdot (1 - bs_{rsm}), \quad \forall t \in T, s \in S_t, m \in M, r \in STOP_{tm} \tag{33}$$

This constraint is part of the overlap case logic. When $bs_{rsm} = 1$, it forces the slot to start at the stoppage start time, aligning the slot to the stoppage window.

$$te_{sm} \leq se_r + BigM \cdot (1 - bs_{rsm}), \quad \forall t \in T, s \in S_t, m \in M, r \in STOP_{tm} \tag{34}$$

$$te_{sm} \geq se_r - BigM \cdot (1 - bs_{rsm}), \quad \forall t \in T, s \in S_t, m \in M, r \in STOP_{tm} \tag{35}$$

These two constraints complete the overlap-case logic. When $bs_{rsm} = 1$, they force the slot end time to be exactly equal to the stoppage end time se_r. Together with constraints (32) and (33), this makes the stoppage slot coincide exactly with the stoppage interval $[ss_r,\ se_r]$.

$$ls_{rsm} + rs_{rsm} + bs_{rsm} = 1, \forall t \in T, s \in S_t, m \in M, r \in STOP_{tm} \tag{36}$$

This constraint enforces that exactly one of the three relative-position cases holds for every slot and stoppage pair. A slot must either occur before the stoppage, after the stoppage, or overlap the stoppage, but never multiple at once.

$$p_{ism} \leq BigM \cdot (1 - bs_{rsm}), \quad \forall t \in T, s \in S_t, m \in M, r \in STOP_{tm}, i \in EIM_m \tag{37}$$

This constraint prevents production in a slot that overlaps a stoppage. When $bs_{rsm} = 1$, the right-hand side becomes zero, forcing $p_{ism} = 0$ for all items in that slot.

$$st_{sm} \leq BigM \cdot (1 - bs_{rsm}), \quad \forall t \in T, s \in S_t, m \in M, r \in STOP_{tm} \tag{38}$$

This constraint prevents setup in a slot that overlaps a stoppage. When $bs_{rsm} = 1$, the right-hand side becomes zero, forcing $st_{sm} = 0$ for all items in that slot.

Each stoppage is represented by exactly one zero-production slot:

$$\sum_{s \in S_t} bs_{rsm} = 1, \quad \forall t \in T, m \in M, r \in STOP_{tm} \tag{39}$$

This constraint ensures that every stoppage interval is explicitly embedded into the consecutive slot sequence.

During a stoppage slot, production and setup activities are forced to zero. This does not permit an unintended sequence-dependent changeover to be hidden inside the stoppage. The transition variables in Constraints (13)–(16) link the item state of each slot to that of the immediately preceding slot, and the corresponding sequence-dependent setup time is imposed whenever the selected item changes. Furthermore, the stoppage constraint forces the setup time of the overlapping slot to zero. Consequently, if the transition from item i to a different item j requires a positive setup time, such a state change cannot occur within the stoppage slot because it would simultaneously require a positive setup time and a zero setup time. The item state is therefore implicitly retained across the stoppage for every transition having a positive sequence-dependent setup requirement. If an off-diagonal transition has zero setup time in the input data, the model may permit that transition because it is operationally defined as a zero-setup change and is therefore not an omitted setup.

For example, suppose item A is selected before a stoppage and the setup time from A to B is 0.5 h. Selecting B as the state of the stoppage slot would activate the A→B transition and require a 0.5 h setup, whereas the stoppage constraint requires setup time to equal zero. These conditions are incompatible; hence, the stoppage slot cannot be used to change from A to B. If B is selected in the first productive slot after the stoppage, the A→B transition is then activated and the 0.5 h setup is incurred normally. Thus, a stoppage cannot provide a zero-cost route around a required sequence-dependent setup.

The stoppage intervals for a given machine and period are assumed to be non-overlapping. If two stoppage intervals overlap or are directly adjacent, they should be merged into a single stoppage interval before model generation. Each stoppage interval must lie within the relevant period time window and must be compatible with the applicable shift-time restrictions.

3.6.4. Mold maintenance

$$b_{ksm} = 0, \quad \forall k \in K, t \in T, s \in S_t, m \in EMK_k \mid mm_{kt} = 1 \tag{40}$$

This constraint prevents mold k from being assigned to any eligible machine in any subperiod $s \in S_t$ when the mold is under maintenance in period t. Since production using a mold is linked to mold assignment, this restriction also prevents production using that mold during maintenance periods.

3.6.5. Holidays

$$st_{sm} = 0, \quad \forall t \in T, s \in S_t, m \in M \mid v_t = 1 \tag{41}$$

This constraint forces setup time to be zero on holidays. Since no production activity is allowed on holidays, setup time cannot be incurred.

$$p_{ism} = 0, \quad \forall t \in T, s \in S_t, m \in M, i \in EIM_m \mid v_t = 1 \tag{42}$$

This constraint enforces zero production on holidays. For any holiday period t, all slot-level production quantities must be set to zero.

$$y_{ism} = y_{i,s-1,m}, \quad \forall t \in T, s \in S_t, m \in M, i \in EIM_m \mid v_t = 1 \tag{43}$$

This constraint preserves the production-selection state across consecutive slots on holidays. It ensures the item selection does not change even though production is halted, matching the original model's logic.

During zero-production slots, production quantities and setup times are forced to zero and mold-assignment variables do not represent active production. On holidays, Constraint (43) explicitly preserves the item-selection state. For stoppage slots, the transition and setup constraints described in Section 3.6.3 prevent any item-state change requiring a positive sequence-dependent setup from occurring within the zero-setup stoppage interval. Accordingly, zero-production mold assignments are interpreted only as auxiliary configuration variables and not as productive mold usage.

The interpretation of the mold-assignment variable is different for productive and nonproductive slots. In a productive slot, the formulation directly links positive production to the selected item and to a mold belonging to the corresponding item–machine compatibility structure. Thus, a positive quantity of item i cannot be produced using a mold that is incompatible with item i or with machine m. The mold-exclusivity and single-mold constraints further ensure that the same physical mold is not simultaneously used by multiple machines and that a machine does not actively use multiple molds in the same slot. Hence, the required item–mold–machine consistency is enforced whenever production is positive.

In a zero-production slot, such as a stoppage or holiday slot, the mold-assignment variable has no productive interpretation because all production quantities are zero. A nonzero mold-assignment value in such a slot is therefore an auxiliary configuration value of the slot-based formulation and is not interpreted as evidence that the mold is physically producing an item. Importantly, mold-maintenance and mold-exclusivity restrictions remain applicable to these variables, so an unavailable mold cannot be used to create production or circumvent a maintenance restriction. For reporting and operational interpretation, item–mold assignments are treated as active production assignments only when the corresponding production quantity is positive.

For example, suppose item A can be produced with mold K1 on machine M1, whereas mold K2 is compatible with M1 but not with item A. If production of A in the slot is positive, the production–mold linking constraints require assignment of a mold from A's eligible set, so K2 cannot support that production. If production is zero, a mold-assignment variable does not generate production and is not

interpreted as a physical item–mold production assignment. Thus, the absence of a separate y-to-b equality in zero-production slots does not enlarge the set of feasible productive schedules.

3.6.6. Item grouping optimization

$$\sum_{h\in HI_i}\sum_{t\in T} i\,a_{ith} \le I0_i, \forall i \in I \tag{44}$$

This constraint limits the quantity of initial inventory allocated to header-level demand. The total inventory allocated across all associated headers and demand periods cannot exceed the initial inventory available for item i. Any unallocated portion of the initial inventory remains available as ordinary item-level inventory in the inventory-balance structure.

$$ia_{ith} = 0, \quad \forall h \in HC, t \in T, i \in IH_h \mid IHD_{hti} = 0 \tag{45}$$

This constraint prevents initial inventory from being allocated to header-period combinations with zero demand.

$$ia_{ith} \le IHD_{hti}, \quad \forall h \in HC, t \in T, i \in IH_h \tag{46}$$

This constraint further limits the allocated initial inventory so that the allocation to a header-period combination does not exceed the corresponding header-level item demand.

$$\sum_{s\in S_\tau}\sum_{m\in EMI_i} p_{ism} \ge \sum_{h\in HI_i}\sum_{t\in T} p\,a_{it\tau h}, \forall i \in I, \tau \in T \tag{47}$$

This constraint links physical production of item i in period τ to the portion of that production allocated to different headers and demand periods. It ensures that the total quantity allocated to header-level demand cannot exceed the physical production quantity of item i in period τ. Any unallocated production remains available as ordinary item-level inventory through the inventory-balance structure.

$$\sum_{h\in HI_i} o\,a_{ith} = O_{it}, \forall i \in I, t \in T \tag{48}$$

This constraint ensures that total outsourced quantity for item i in period t is distributed across headers consistently. The sum of header-level outsourcing allocations equals the item-level outsourcing decision O_{it}. This equality is retained because subcontracted quantities are assumed to be used only for satisfying known header-level demand in the corresponding item-period combination.

Constraints (44), (47), and (48) provide the material-conservation link between the item-level inventory balance and the header-level fulfillment variables. The item-level inventory–backlog equation remains the physical material-balance equation: initial inventory, in-house production, and subcontracting constitute supply, while aggregate item demand, ending inventory, and backlog constitute its disposition. The header-level variables do not introduce additional physical supply or additional demand into this balance. Instead, they act as allocation variables that identify which already-conserved units are credited toward individual grouped customer demands.

Constraint (44) prevents double allocation of initial inventory by requiring the sum of all header-level allocations originating from initial inventory to remain within the available initial inventory. Similarly, Constraint (47) prevents double allocation of production from any production period τ: The

total quantity from that production period credited across all relevant headers and demand periods cannot exceed the physical quantity actually produced in period τ. Constraint (48) performs the corresponding reconciliation for outsourcing by requiring the total header-level outsourcing allocation to equal the item-level outsourcing quantity. Therefore, a physical unit can be attributed to header fulfillment only within the quantity already available in the corresponding item-level source.

The inequality in Constraint (47) is intentional rather than an omission. Not every unit produced in period τ must necessarily be allocated immediately to a specific header. Production in excess of current header allocations may remain as ordinary item-level inventory and subsequently satisfy later demand. Replacing the inequality by equality would incorrectly force all production to be assigned immediately to header demand and would prevent legitimate advance-production and inventory decisions.

Consider one item with total demand of 100 units comprising two headers, H1 = 60 units and H2 = 40 units. Suppose 20 units are available as initial inventory, 70 units are produced in-house, and 10 units are subcontracted. A feasible allocation is H1 = 20 units from initial inventory + 40 units from production, and H2 = 30 units from production + 10 outsourced units. The header allocations therefore satisfy $20 \leq 20$ for initial inventory, $40 + 30 = 70 \leq 70$ for production, and $10 = 10$ for outsourcing. Simultaneously, the physical item balance is $20 + 70 + 10 - 100 = 0$. No unit is counted twice: The item-level equation conserves the material physically, while the header-level variables only identify how that same conserved material satisfies H1 and H2. If instead 80 units were produced and only 70 were presently allocated to headers, Constraint (47) would permit the remaining 10 units to remain as item-level inventory for subsequent demand.

$$ia_{ith} + \sum_{\tau_1 \in T | \tau_1 \leq \tau} p\, a_{it\tau_1 h} + oa_{ith} \geq IHD_{hti}\, hf_{ht\tau} \quad \forall h \in HC, t \in T, i \in IH_h, \tau \in T \mid \tau \geq t, IHD_{hti} > 0 \tag{49}$$

This constraint enforces satisfaction of header-level item demand using three sources: allocated inventory, allocated in-house production up to the completion period, and allocated outsourcing. This constraint ensures that if header h demand for item i in period t is declared completed in period τ, then the total quantity allocated from initial inventory, in-house production available up to period τ, and outsourcing must be at least equal to the corresponding header-level item demand. If $hf_{ht\tau} = 0$, the right-hand side becomes zero and the constraint does not force completion-related supply for that candidate completion period.

This formulation permits production completed before the demand period t to be allocated to the corresponding header demand, representing advance production or inventory-based fulfillment. As such, if advance allocation is not allowed, the summation can be restricted to periods τ1 satisfying $t \leq \tau 1 \leq \tau$.

$$\sum_{\tau \in T | \tau \geq t} h\, f_{ht\tau} = 1, \forall h \in HC, t \in T \mid \sum_{i \in IH_h} I\, HD_{hti} > 0 \tag{50}$$

This constraint forces a unique completion period τ to be chosen for each header demand period t when that header has positive demand. Exactly one of the $hf_{t\tau}$ variables must be 1 over the feasible completion periods.

$$pa_{it\tau h} \le IHD_{hti} \sum_{\rho \in T \mid \rho \ge \tau} h f_{ht\rho} \quad \forall h \in HC, t \in T, i \in IH_h, \tau \in T \mid \tau \ge t, IHD_{hti} > 0 \tag{51}$$

This ensures production allocated to a demand can occur only no later than the chosen completion period.

$$pa_{it\tau h} = 0, \quad \forall h \in HC, t \in T, i \in IH_h, \tau \in T \mid IHD_{hti} = 0 \tag{52}$$

This constraint prevents allocating production toward header-demand pairs that do not exist. If $IHD_{hti} = 0$, then the associated allocation variables must be zero.

$$oa_{ith} = 0, \quad \forall h \in HC, t \in T, i \in IH_h \mid IHD_{hti} = 0 \tag{53}$$

This constraint similarly prevents allocating outsourcing to header-demand pairs with zero demand. It maintains consistency so that only demanded combinations can receive outsourced quantity.

$$pa_{it\tau h} = 0, \forall k \in K, t \in T, \tau \in T, i \in EIK_k, h \in HI_i \mid mmo_k = 1, \left(\text{card}(EKI_i) - \sum_{k_1 \in EKI_i} m m_{k_1,\tau} \right) = 0 \tag{54}$$

This constraint blocks certain header-level production allocations under the mold-maintenance grouping option captured by mmo_k. The condition is preserved exactly as stated in the original model, and it enforces the intended stoppage / maintenance interaction at the allocation level. When the maintenance-grouping option is active, header demand in period t is not allowed to be fulfilled through production allocation if all molds eligible for item i are under maintenance in that demand period. This is a business-rule restriction and applies even if advance production would otherwise be available.

3.6.7. Variable domains

The binary decision variables are defined as follows:

$$y_{ism}, z_{ijsm}, b_{ksm}, ls_{rsm}, rs_{rsm}, bs_{rsm}, hf_{ht\tau} \in \{0,1\}$$

The nonnegative continuous decision variables are defined as follows:

$$p_{ism}, pk_{ismk}, I_{it}, B_{it}, O_{it}, ia_{ith}, pa_{it\tau h}, oa_{ith}, n_{sm}, st_{sm}, ts_{sm}, te_{sm} \ge 0$$

4. Case study and computational investigation

In this section, we report the industrial case context and the computational investigation used to evaluate the proposed mixed-integer programming formulation. The study comprises 97 completed experiments organized to examine dimensional scalability, sparse model construction, solver behavior, operational response, and managerial implications. Before numerical analysis, each run was checked for a feasible incumbent, consistency between the declared and realized instance dimensions, and satisfaction of the independent validation procedure. The resulting evidence base supports a systematic

assessment of computational performance and operational behavior across scale, sensitivity, stress, and controlled validation settings.

4.1. Industrial setting, implementation environment, and verification protocol

The case study is based on the injection-molding operations of an established kitchen-appliance manufacturer producing a diversified portfolio of consumer-durable products. All 97 computational instances used in this study were obtained from the case-study company and reflect its production data, resource structure, compatibility relationships, and planning conditions. The company has 24 injection-molding machines in the full production system. Small and Medium instances correspond to specific production lines within the plant, whereas Large instances represent the full production line. When one or more machines are unavailable because of planned machine maintenance, those machines are excluded from the available machine set before schedule generation; consequently, the Large instances contain 21–24 active machines. Machine maintenance is therefore handled as an availability preprocessing step, with only machines available for the relevant planning run included in the schedulable machine set. The production environment is further characterized by high product variety, repeated mold changes, shared use of physical molds across eligible machines, heterogeneous cycle times and cavity counts, time-phased demand, planned stoppages, holidays, mold maintenance, subcontracting options, and grouped customer-level demand.

The economic and penalty parameters are treated as exogenous deterministic planning inputs rather than parameters estimated from the optimization results. In-house production cost, subcontracting cost, and inventory carrying cost represent the economic inputs associated with the corresponding case-study planning data, while the penalty coefficients represent planning priorities: The sourcing-preference penalty discourages outsourcing of items designated for in-house production, the grouped-demand delay penalty discourages late completion of customer-level demand, and the backlog penalty discourages unmet demand across periods. All coefficients are fixed before each optimization run. Because the company's absolute monetary data are commercially confidential, monetary performance is reported using a common scaling convention where required; this transformation does not alter relative cost comparisons or the resulting percentage improvements. Rather than arbitrarily perturbing confidential monetary coefficients, the robustness study varies the principal operational drivers, machines, items, molds, and planning periods, and separately examines high setup intensity, tooling scarcity, operational disruption, and demand pressure. The realized solutions provide an additional check on the penalty architecture: only 0.636% of modeled demand is subcontracted, while no backlog or grouped-demand delay penalty is incurred across the 97 optimized cases, indicating that the reported schedules are not being produced through systematic reliance on penalized recourse decisions.

The optimization system was implemented in C#.NET and solved through the native .NET interface of Gurobi Optimizer. All experiments were executed on a 64-bit Windows workstation equipped with a 12th Gen Intel Core i9-12900K processor and 32 GB RAM. Eight solver threads were assigned to each sequential run. The relative MIP-gap tolerance was 10^{-4}, and the overall time limit was 3600 s. In addition to this time limit, a rounded-objective stagnation rule was used: A run could be terminated when the rounded incumbent objective failed to improve for 60 s in Small and controlled cases, 180 s in Medium cases, or 600 s in Large cases. The native Gurobi status and the application-level stagnation reason were recorded separately.

For every run, the evaluation covered the realized dimensions, compatibility profile, solver limits, model statistics, solution-quality measures, objective decomposition, production schedule, tooling assignments, inventory and backlog trajectory, and grouped-demand completion. The post-solve verifier independently checks temporal boundaries, shift continuity, machine–item eligibility, mold–item and mold–machine compatibility, mold exclusivity, transition consistency, setup-time calculations, production capacity, stoppage, holiday and maintenance restrictions, inventory–backlog conservation, subcontracting consistency, grouped-demand allocation, objective reconstruction, and KPI consistency.

The verification protocol produced a complete and internally consistent evidence base. All 97 experiment runs produced feasible solutions and satisfied the independent validation checks. Ninety-six runs reached the Gurobi OPTIMAL status. One Medium sensitivity instance, SENS_ITEMS_HIGH_02, terminated under the rounded-objective stagnation rule with a final relative gap of 0.00450084%, which is below 0.01%. This case is reported with its actual interrupted solver status and high-quality incumbent. Detailed solver-performance analysis is presented in Section 4.4; the important result at this stage is that every completed instance yields a feasible and independently validated solution.

4.2. Experimental design and instance taxonomy

The computational program was organized into four complementary blocks. The first block contains scale benchmarks designed to expose the growth of the integrated formulation over progressively larger machine, item, mold, and time-horizon dimensions. The second block contains designed two-level sensitivity experiments for machines, items, molds, and planning periods. The third block subjects the system to operational stress through high setup intensity, tooling scarcity, operational disruption, and demand pressure. The fourth block contains controlled mathematical validation cases, based on the case-study data structure, that isolate specific model mechanisms under transparent conditions. This architecture separates three scientific questions: Whether the formulation scales, how model behavior changes when principal dimensions are varied, and whether the formulation responds correctly when specific operational mechanisms are activated.

The scale-benchmark block comprises 10 Small, 10 Medium, and 25 Large instances. Small benchmarks represent specific production-line settings with five machines, 15–20 items, 10–15 molds, 5–7 periods, and 4–5 slots per period. Medium benchmarks represent larger production-line settings with 11–15 machines, 20–39 items, 15–30 molds, 7–10 periods, and 4–5 slots per period. Large benchmarks represent the company's full injection-molding production line and contain 21–24 active machines, 40–80 items, 32–50 molds, 7–10 periods, and 4–5 slots per period. The full plant has 24 machines; the observed Large-instance range of 21–24 results from excluding machines that are unavailable because of planned maintenance before scheduling. The sensitivity block adds four Small machine-low cases, 24 Medium cases associated with item, mold, and period levels, and four Large machine-high cases. The stress block is Large because its purpose is to observe the interaction between difficult operating conditions and full-line model structures. The controlled validation cases use five machines, 12 items, 8 molds, 5 periods, and 4 slots per period so that individual logical mechanisms can be checked without confounding scale effects.

Table 1. Experimental design and distribution of the 97 case-study instances.

Experimental block	Completed runs	Class allocation	Primary scientific purpose	Design structure
Scale benchmarks	45	10 Small; 10 Medium; 25 Large	Quantify dimensional and model-size scalability	Independent instances across class-specific ranges
Sensitivity analysis	32	4 Small; 24 Medium; 4 Large	Evaluate designed contrasts in machines, items, molds, and periods	4 factors × 2 levels × 4 replications
Operational stress	12	12 Large	Evaluate setup, tooling, disruption, and demand stress	4 stress scenarios × 3 replications
Mathematical validation	8	8 controlled	Verify specific constraint groups and business rules	4 controlled cases × 2 replications
Complete executed programme	97	14 Small; 34 Medium; 41 Large; 8 controlled	Integrated empirical evidence base	97 completed, independently validated runs

The final computational design therefore contains 97 completed case-study instances: 45 scale benchmarks, 32 sensitivity experiments, 12 operational-stress experiments, and 8 controlled validation cases. The experiment count and class allocation are summarized in Table 1. All aggregate quantities, medians, ranges, and proportions reported below are calculated over the corresponding completed runs.

4.3. Model dimensionality and effectiveness of sparse construction

The integrated formulation contains several variable families whose natural dense domains grow multiplicatively with the number of machines, items, molds, periods, and slots. The most prominent examples are machine–item production variables, machine-specific item-transition variables, mold-assignment variables, and mold-level production variables indexed by feasible machine–item–mold triples. A dense implementation would create these variables over large Cartesian products and then force physically impossible combinations to zero. The implemented formulation instead constructs variables only on verified eligibility sets: E(MI) for machine–item pairs, E(KI) for mold–item pairs, E(KM) for mold–machine pairs, E(MIK) for feasible machine–item–mold triples, and E(TR) for machine-specific item-transition arcs. Sparse construction therefore changes the computational representation, not the underlying feasible production decisions.

Across the 97 completed instances, the median machine–item density is 52.16%, the median mold–item density is 27.79%, and the median mold–machine density is 54.08%. The more restrictive three-way machine–item–mold density has a median of only 8.07% of the full Cartesian product, while feasible transition arcs occupy a median of 27.71% of the corresponding machine–item–item domain. These values explain why eligibility cannot be treated as a minor preprocessing detail: The three-way tooling structure removes the majority of combinations that have no physical interpretation, while retaining at least one feasible mold route for every eligible machine–item pair.

Table 2. Scale benchmark characteristics and solution outcomes for all 45 case-study instances.

Instance	Class / design	M/I/K/P/S	Variables	Constraints	Objective value	Best bound	Time (s)	MIP gap (%)
SCALE_S_01	Small	5/19/13/7/5	33,664	22,103	67,239.36	67,239.36	0.997	0.0000
SCALE_S_02	Small	5/17/13/5/4	16,604	11,463	38,358.59	38,358.59	0.752	0.0000
SCALE_S_03	Small	5/18/10/6/4	22,644	15,234	52,299.94	52,299.94	1.275	0.0000
SCALE_S_04	Small	5/20/10/5/4	23,967	13,957	45,715.61	45,711.25	2.156	0.0095
SCALE_S_05	Small	5/16/10/5/5	22,199	14,926	37,228.66	37,228.66	0.364	0.0000
SCALE_S_06	Small	5/17/13/6/5	27,736	18,032	43,624.08	43,624.08	0.320	0.0000
SCALE_S_07	Small	5/16/10/7/5	29,454	21,072	48,845.06	48,845.06	0.373	0.0000
SCALE_S_08	Small	5/18/15/6/5	28,404	18,048	49,400.04	49,400.04	0.363	0.0000
SCALE_S_09	Small	5/19/12/7/4	32,188	19,164	57,963.66	57,958.47	2.901	0.0090
SCALE_S_10	Small	5/15/12/6/4	17,443	12,928	47,681.33	47,681.33	0.466	0.0000
SCALE_M_01	Medium	11/20/29/9/5	161,743	114,896	85,472.88	85,472.88	11.864	0.0000
SCALE_M_02	Medium	13/21/26/8/5	132,195	97,289	81,698.23	81,698.23	2.787	0.0000
SCALE_M_03	Medium	14/33/15/9/4	211,420	116,311	134,255.13	134,255.13	14.821	0.0000
SCALE_M_04	Medium	15/23/19/7/5	104,283	77,512	76,365.71	76,365.71	1.558	0.0000
SCALE_M_05	Medium	11/23/16/7/5	153,474	99,263	83,821.00	83,821.00	4.890	0.0000
SCALE_M_06	Medium	12/21/30/8/5	144,478	102,617	85,628.09	85,628.09	2.912	0.0000
SCALE_M_07	Medium	12/30/19/10/5	284,473	173,833	134,971.81	134,971.81	37.029	0.0000
SCALE_M_08	Medium	13/20/24/8/4	87,981	69,977	64,328.17	64,328.17	1.346	0.0000
SCALE_M_09	Medium	14/32/28/9/4	217,699	124,131	131,985.42	131,985.42	7.558	0.0000
SCALE_M_10	Medium	11/39/19/8/4	344,969	147,523	150,864.85	150,864.85	114.412	0.0000
SCALE_L_01	Large	22/59/41/10/4	854,783	374,147	273,695.97	273,695.97	227.751	0.0000
SCALE_L_02	Large	21/63/50/7/4	668,212	273,218	204,974.60	204,974.60	122.235	0.0000
SCALE_L_03	Large	21/45/35/8/4	414,994	226,400	164,402.85	164,402.85	17.589	0.0000
SCALE_L_04	Large	23/59/43/7/4	549,809	243,568	188,348.35	188,348.35	69.135	0.0000
SCALE_L_05	Large	23/40/49/9/4	363,837	212,713	175,963.71	175,963.71	24.567	0.0000
SCALE_L_06	Large	24/63/34/10/5	1,055,104	435,765	305,278.58	305,278.58	279.085	0.0000
SCALE_L_07	Large	21/45/40/8/4	471,601	247,087	174,329.02	174,329.02	54.042	0.0000
SCALE_L_08	Large	21/75/38/9/5	1,492,923	526,838	327,129.62	327,123.95	531.668	0.0017
SCALE_L_09	Large	22/75/34/10/5	1,595,377	576,357	371,322.45	371,322.45	537.505	0.0000
SCALE_L_10	Large	23/57/40/7/4	533,138	232,660	185,681.98	185,681.98	53.764	0.0000
SCALE_L_11	Large	23/80/46/9/5	1,614,714	565,022	350,784.10	350,784.10	520.045	0.0000
SCALE_L_12	Large	21/76/43/9/5	1,416,302	513,372	317,315.32	317,315.32	593.548	0.0000
SCALE_L_13	Large	22/64/48/9/4	891,901	358,641	261,934.28	261,934.28	256.479	0.0000
SCALE_L_14	Large	23/64/44/10/4	924,101	376,839	316,360.67	316,360.67	210.742	0.0000
SCALE_L_15	Large	23/46/50/7/5	462,176	239,339	161,436.03	161,436.03	12.083	0.0000
SCALE_L_16	Large	21/42/47/8/5	503,984	270,837	159,100.70	159,100.70	19.956	0.0000

Continued on next page

Instance	Class / design	M/I/K/P/S	Variables	Constraints	Objective value	Best bound	Time (s)	MIP gap (%)
SCALE_L_17	Large	21/64/32/9/4	892,902	363,518	277,691.58	277,684.35	286.249	0.0026
SCALE_L_18	Large	21/40/39/8/4	372,512	204,912	147,145.64	147,145.64	15.615	0.0000
SCALE_L_19	Large	22/40/35/8/4	329,614	192,730	157,006.12	157,006.12	8.574	0.0000
SCALE_L_20	Large	23/77/32/8/4	1,020,779	368,189	282,886.10	282,874.45	474.467	0.0041
SCALE_L_21	Large	24/73/50/9/5	1,266,967	487,670	312,990.99	312,990.99	404.465	0.0000
SCALE_L_22	Large	21/54/35/7/4	504,722	240,330	187,827.65	187,827.65	65.918	0.0000
SCALE_L_23	Large	22/43/40/8/5	450,508	250,418	161,673.46	161,673.46	71.564	0.0000
SCALE_L_24	Large	23/80/37/8/5	1,346,602	474,296	289,713.35	289,713.35	1176.581	0.0000
SCALE_L_25	Large	24/61/43/9/4	747,436	331,419	266,071.67	266,071.67	129.249	0.0000

Table 3. Sensitivity-analysis instance characteristics and solution outcomes for all 32 case-study instances.

Instance	Class / design	M/I/K/P/S	Variables	Constraints	Objective value	Best bound	Time (s)	MIP gap (%)
SENS_MACHINES_LOW_01	Small; Machines-Low	5/20/12/6/5	34,929	21,724	54,679.30	54,679.30	2.086	0.0000
SENS_MACHINES_LOW_02	Small; Machines-Low	5/15/10/6/4	19,222	13,579	47,557.76	47,557.76	0.966	0.0000
SENS_MACHINES_LOW_03	Small; Machines-Low	5/19/10/5/4	19,788	13,791	49,902.25	49,902.25	1.872	0.0000
SENS_MACHINES_LOW_04	Small; Machines-Low	5/15/12/6/4	15,519	11,580	52,726.16	52,726.16	0.593	0.0000
SENS_MACHINES_HIGH_01	Large; Machines-High	24/80/46/9/4	1,164,281	410,977	349,633.02	349,618.10	890.793	0.0043
SENS_MACHINES_HIGH_02	Large; Machines-High	24/62/31/10/5	1,060,951	469,610	285,309.81	285,309.81	311.351	0.0000
SENS_MACHINES_HIGH_03	Large; Machines-High	24/44/37/8/4	364,354	204,194	158,644.92	158,644.92	14.043	0.0000
SENS_MACHINES_HIGH_04	Large; Machines-High	24/74/35/9/5	1,358,481	489,180	288,922.27	288,922.27	294.273	0.0000
SENS_ITEMS_LOW_01	Medium; Items-Low	15/20/16/10/5	125,467	99,992	102,763.61	102,763.61	3.119	0.0000
SENS_ITEMS_LOW_02	Medium; Items-Low	10/20/20/7/4	98,917	67,450	65,959.76	65,959.76	1.852	0.0000
SENS_ITEMS_LOW_03	Medium; Items-Low	11/20/24/9/5	175,296	124,674	97,135.38	97,125.67	32.960	0.0100
SENS_ITEMS_LOW_04	Medium; Items-Low	13/20/22/9/5	139,827	103,915	83,510.65	83,510.65	3.188	0.0000

Continued on next page

Instance	Class / design	M/I/K/P/S	Variables	Constraints	Objective value	Best bound	Time (s)	MIP gap (%)
SENS_ITEMS_HIGH_01	Medium; Items-High	15/40/26/10/4	322,263	166,993	194,550.17	194,538.71	37.323	0.0059
SENS_ITEMS_HIGH_02	Medium; Items-High	10/40/23/10/4	459,145	210,151	185,318.39	185,310.05	510.873	0.0045
SENS_ITEMS_HIGH_03	Medium; Items-High	11/40/27/7/5	370,473	171,229	121,345.54	121,345.54	65.695	0.0000
SENS_ITEMS_HIGH_04	Medium; Items-High	13/40/25/8/5	384,889	181,887	147,309.90	147,309.90	41.693	0.0000
SENS_MOLDS_LOW_01	Medium; Molds-Low	14/32/15/9/4	183,475	118,559	150,330.69	150,330.69	16.656	0.0000
SENS_MOLDS_LOW_02	Medium; Molds-Low	15/22/15/7/4	85,101	66,539	79,149.38	79,149.38	1.298	0.0000
SENS_MOLDS_LOW_03	Medium; Molds-Low	15/40/15/8/4	261,712	140,248	153,189.08	153,189.08	31.491	0.0000
SENS_MOLDS_LOW_04	Medium; Molds-Low	11/38/15/9/4	374,241	167,587	169,664.67	169,659.30	110.923	0.0032
SENS_MOLDS_HIGH_01	Medium; Molds-High	12/29/30/10/5	299,276	176,704	155,444.58	155,444.58	46.129	0.0000
SENS_MOLDS_HIGH_02	Medium; Molds-High	13/40/30/8/5	370,524	170,447	152,271.85	152,271.85	52.181	0.0000
SENS_MOLDS_HIGH_03	Medium; Molds-High	14/31/30/9/4	222,865	126,142	131,730.88	131,730.88	20.749	0.0000
SENS_MOLDS_HIGH_04	Medium; Molds-High	14/28/30/7/4	134,694	82,876	88,119.76	88,119.76	2.591	0.0000
SENS_PERIODS_LOW_01	Medium; Periods-Low	14/40/25/7/4	233,666	120,112	131,492.63	131,492.63	58.576	0.0000
SENS_PERIODS_LOW_02	Medium; Periods-Low	15/31/29/7/5	199,497	110,530	94,839.00	94,839.00	3.666	0.0000
SENS_PERIODS_LOW_03	Medium; Periods-Low	10/21/17/7/4	107,155	76,453	70,081.94	70,081.94	11.041	0.0000
SENS_PERIODS_LOW_04	Medium; Periods-Low	12/40/15/7/4	263,147	135,001	149,405.02	149,405.02	78.917	0.0000
SENS_PERIODS_HIGH_01	Medium; Periods-High	13/37/29/10/4	328,380	155,863	172,434.97	172,434.97	29.882	0.0000
SENS_PERIODS_HIGH_02	Medium; Periods-High	13/27/18/10/4	201,003	132,322	119,746.21	119,746.21	32.498	0.0000
SENS_PERIODS_HIGH_03	Medium; Periods-High	14/39/22/10/5	433,388	222,596	168,203.82	168,203.82	83.019	0.0000
SENS_PERIODS_HIGH_04	Medium; Periods-High	10/37/20/10/5	439,394	216,253	182,461.37	182,461.37	81.981	0.0000

Table 4. Operational-stress instance characteristics and solution outcomes for all 12 case-study instances.

Instance	Class / design	M/I/K/P/S	Variables	Constraints	Objective value	Best bound	Time (s)	MIP gap (%)
STRESS_DEMANDPRESSURE_01	Large; Demand pressure	21/79/37/8/4	1,194,243	398,709	482,015.86	481,998.38	1279.001	0.0036
STRESS_DEMANDPRESSURE_02	Large; Demand pressure	21/61/43/9/4	830,844	341,575	436,587.90	436,555.79	238.440	0.0074
STRESS_DEMANDPRESSURE_03	Large; Demand pressure	21/56/34/7/4	579,814	250,167	299,785.74	299,785.74	79.892	0.0000
STRESS_HIGHSETUPINTENSITY_01	Large; High setup intensity	21/55/42/7/4	519,064	242,304	185,174.34	185,174.34	89.301	0.0000
STRESS_HIGHSETUPINTENSITY_02	Large; High setup intensity	23/62/37/10/4	860,190	369,269	301,292.55	301,292.55	323.537	0.0000
STRESS_HIGHSETUPINTENSITY_03	Large; High setup intensity	24/44/43/7/5	403,220	223,176	132,654.11	132,654.11	18.056	0.0000
STRESS_OPERATIONALDISRUPTION_01	Large; Operational disruption	22/79/49/8/4	1,103,367	391,200	319,780.41	319,767.97	597.366	0.0039
STRESS_OPERATIONALDISRUPTION_02	Large; Operational disruption	24/75/46/9/4	1,072,640	406,085	325,121.04	325,097.00	312.176	0.0074
STRESS_OPERATIONALDISRUPTION_03	Large; Operational disruption	24/56/31/10/5	855,279	409,779	274,327.05	274,327.05	130.002	0.0000
STRESS_TOOLINGSCARCITY_01	Large; Tooling scarcity	21/40/40/8/5	42,946	38,154	156,768.87	156,768.87	1.718	0.0000
STRESS_TOOLINGSCARCITY_02	Large; Tooling scarcity	21/62/46/9/4	60,225	47,597	288,776.66	288,749.47	19.036	0.0094
STRESS_TOOLINGSCARCITY_03	Large; Tooling scarcity	22/44/31/7/4	29,460	28,172	156,385.75	156,385.75	0.343	0.0000

Table 5. Controlled mathematical-validation instance characteristics and solution outcomes for all 8 case-study instances.

Instance	Class / design	M/I/K/P/S	Variables	Constraints	Objective value	Best bound	Time (s)	MIP gap (%)
VALIDATION_HEADERCOMPLETIONANDBACKLOG_01	Controlled; Header completion and backlog	5/12/8/5/4	12,426	9,610	25,898.16	25,898.16	0.268	0.0000
VALIDATION_HEADERCOMPLETIONANDBACKLOG_02	Controlled; Header completion and backlog	5/12/8/5/4	11,223	9,027	31,486.26	31,486.26	0.173	0.0000
VALIDATION_MAINTENANCEHOLIDAYALTERNATIVE_01	Controlled; Maintenance and holiday alternatives	5/12/8/5/4	11,043	9,295	26,284.48	26,284.48	0.175	0.0000
VALIDATION_MAINTENANCEHOLIDAYALTERNATIVE_02	Controlled; Maintenance and holiday alternatives	5/12/8/5/4	9,979	9,055	28,551.57	28,551.57	0.103	0.0000
VALIDATION_SHAREDMOLDEXCLUSIVITY_01	Controlled; Shared-mold exclusivity	5/12/8/5/4	9,418	8,402	29,489.69	29,489.69	0.142	0.0000
VALIDATION_SHAREDMOLDEXCLUSIVITY_02	Controlled; Shared-mold exclusivity	5/12/8/5/4	10,321	8,425	32,427.50	32,427.50	0.133	0.0000
VALIDATION_TIMINGANDSHIFTBOUNDARIES_01	Controlled; Timing and shift boundaries	5/12/8/5/4	9,779	8,623	33,576.67	33,576.67	0.220	0.0000
VALIDATION_TIMINGANDSHIFTBOUNDARIES_02	Controlled; Timing and shift boundaries	5/12/8/5/4	12,125	9,469	26,180.08	26,180.08	0.143	0.0000

In Tables 2–5, M/I/K/P/S denotes the numbers of machines, items, molds, periods, and slots per period, respectively. Together, Tables 2–5 report the instance-level dimensions, incumbent objective value, solver best bound, runtime, and final relative MIP gap for all 97 experiments.

Tables 2–5 provide the complete instance-level computational record for the 97 experiments. Across the complete experiment set, model size spans 9,418 to 1,614,714 actual variables, 8,402 to 576,357 constraints, and 35,845 to 5,640,496 nonzero coefficients. The medians are 299,276 variables, 155,863 constraints, and 1,142,198 nonzeros. Relative to the corresponding dense theoretical construction, sparse variable generation reduces the model by a median of 83.00%, with an observed range from 65.21% to 99.05%. The upper extreme occurs in the tooling-scarcity stress design and reflects the very restricted feasible compatibility structure. Within the ordinary Large scale benchmarks, the reduction is consistently between 84.78% and 89.79%.

The class-level growth is substantial. Small scale benchmarks contain 16,604–33,664 variables and 11,463–22,103 constraints. Medium benchmarks increase to 87,981–344,969 variables and 69,977–173,833 constraints. Large benchmarks contain 329,614–1,614,714 variables and 192,730–576,357 constraints. The largest variable model is SCALE_L_11 with 1,614,714 actual variables and

5,640,496 nonzero coefficients, compared with 12,589,805 variables in its dense theoretical counterpart. The largest constraint system is SCALE_L_09 with 576,357 constraints. These results establish that the computational study reaches genuinely large integrated scheduling formulations rather than merely increasing nominal input dimensions.

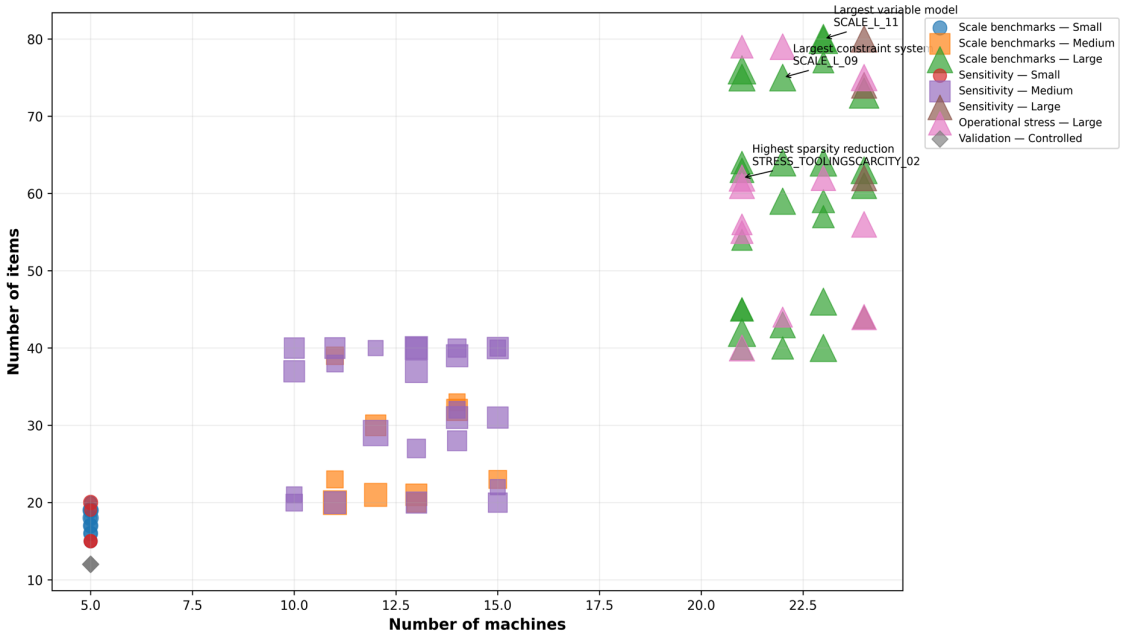

Figure 2. Multidimensional landscape of the 97 executed instances. Position represents machine and item dimensions, bubble area represents mold–period–slot exposure, marker shape represents instance class, and series identify the experimental block.

Figure 2 shows three well-separated dimensional regimes, but it also demonstrates substantial within-class heterogeneity. Instances with similar machine counts can differ markedly in items, molds, periods, and compatibility exposure. Consequently, computational scale cannot be inferred from machine count alone. The number of feasible transition arcs grows approximately with the square of the eligible item set on each machine, whereas mold-level production grows with the number of feasible machine–item–mold triples. This explains why two instances with comparable nominal dimensions may generate very different binary-variable counts and matrix densities. The subsequent computational analysis therefore evaluates runtime and solution quality against model statistics and compatibility structure, rather than treating Small, Medium, and Large as homogeneous categories.

Taken together, the experimental taxonomy and sparse-construction analysis provide a scientifically defensible foundation for the remaining results. The evidence covers controlled microinstances and industrial-scale models exceeding 1.6 million variables; every completed solution passes independent validation; and the exact model-size reduction achieved by sparse indexing is computed for every instance. Sections 4.4–4.9 build on this foundation to examine computational scalability, economic and operational performance, sensitivity, stress response, and managerial implications.

4.4. Computational scalability and solver behavior

The computational evaluation was designed to distinguish three questions that are frequently conflated in mixed-integer scheduling studies: Whether the formulation can generate feasible schedules, whether model growth remains computationally manageable, and which structural characteristics create disproportionate search difficulty. The analysis uses all 97 completed and independently validated instances summarized in Section 4.2 and Tables 2–5. Ninety-six runs terminated with the actual Gurobi status OPTIMAL. The remaining case, SENS_ITEMS_HIGH_02, was terminated by the rounded-objective-stagnation rule with a final relative gap of 0.00450084%. Because the latter is an interrupted rather than an optimal termination, it is reported with its actual interrupted status and high-quality incumbent.

Tables 2–5 report the incumbent objective, solver best bound, runtime, and final relative MIP gap for every instance, while the solver termination status is summarized in the accompanying text. These measures jointly provide transparent evidence of solution quality and enable the reported MIP gaps to be independently interpreted from the incumbent–bound separation.

4.4.1. Analytical protocol

Solver runtime varies from 0.1025 seconds to 1,279.001 seconds, spanning more than four orders of magnitude. Consequently, analyses of model growth and runtime were performed on logarithmic scales. Monotonic associations were quantified using Spearman rank correlations, with 1,500 nonparametric bootstrap replications used to obtain 95% confidence intervals. A descriptive power-law relation was estimated with heteroskedasticity-consistent HC3 standard errors:

$$\log_{10}(R_r) = \alpha + \beta \log_{10}(V_r) + \varepsilon_r$$

where R_r is the Gurobi runtime and V_r is the number of actual sparse variables in instance r. The model is used as a scaling description, not as a causal law or an extrapolation beyond the experimental domain. Variables, constraints, nonzero coefficients, transition arcs, and feasible machine-item-mold triples are highly collinear measures of the same underlying structural expansion. Therefore, a standardized principal-component analysis was also used to construct a latent structural-scale index. Finally, instance-specific residuals from the power-law relation were converted into runtime multipliers. A multiplier greater than one identifies a case that required more time than expected from its sparse variable count alone and therefore provides a direct diagnostic of size-adjusted combinatorial difficulty.

4.4.2. Structural drivers of computational effort

Runtime is strongly associated with every major measure of sparse model expansion. The Spearman correlation between runtime and actual sparse variables is 0.9518 (bootstrap 95% confidence interval 0.9165–0.9701). The corresponding associations are 0.9543 for binary variables, 0.9498 for nonzero coefficients, 0.9450 for transition arcs, 0.9380 for feasible machine-item-mold triples, and 0.9376 for total constraints. These values establish that computational growth is not driven by a single bookkeeping count. It reflects simultaneous expansion of assignment binaries, transition structure, tooling alternatives, and the coefficient matrix.

The fitted power-law model explains 90.26% of the variation in log runtime. The estimated elasticity is $\beta = 1.6087$, with a robust 95% confidence interval of 1.5194–1.6980. Within the tested design, a tenfold increase in sparse variables is therefore associated with an approximately 40.6–fold increase in runtime, with an interval of roughly 33.1–49.9–fold. The relation is superlinear, which is consistent with the fact that model growth simultaneously enlarges the root relaxation, transition network, branching space, and incumbent-improvement opportunities. Figure 3 also shows that the prediction interval widens substantially at the largest scales; hence model size provides a strong expectation but not a deterministic runtime guarantee.

The principal-component analysis reinforces this conclusion while avoiding unstable interpretation of collinear size measures. The first component explains 98.88% of the common variation across actual variables, constraints, nonzeros, transition arcs, and feasible triples, with nearly equal loadings on all five measures. A robust model based on this latent scale component explains approximately 90.0% of log-runtime variation. Once this common scale is accounted for, experimental-block indicators, triple density, period count, and aggregate disruption count do not show stable independent cross-sectional effects. This does not imply that those factors are operationally irrelevant; rather, their operational effects should be examined through the designed sensitivity and stress contrasts in Sections 4.6 and 4.7 rather than inferred from a highly collinear pooled regression.

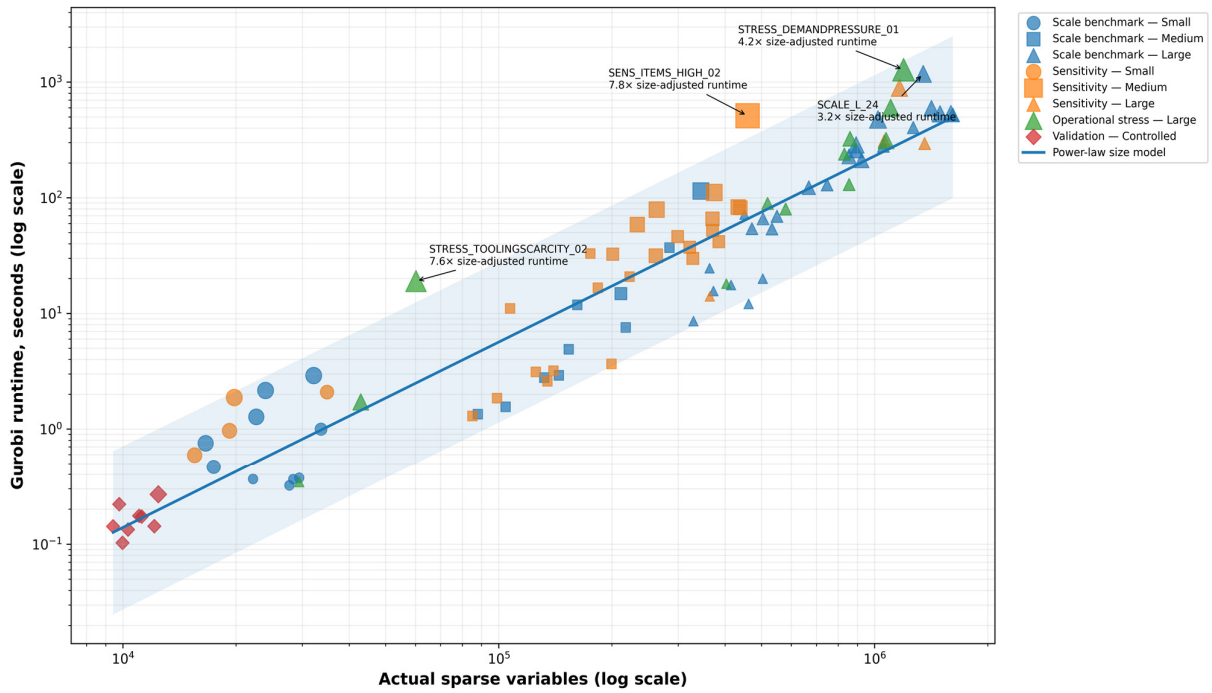

Figure 3. Complexity-performance atlas for all 97 completed experiments. Horizontal and vertical axes are logarithmic; bubble area reflects branch-and-bound node count. The curve is the fitted power-law relation and the shaded region is its 95% prediction interval. Labelled cases exhibit the largest positive size-adjusted runtime residuals.

4.4.3. Runtime scaling across the experimental program

The complete experiment program requires 3.427 hours of Gurobi optimization and 3.526 hours of total measured elapsed time, including model generation, independent validation, and post-solve processing. Median solver runtime across all 97 instances is 20.749 seconds. Thirty-five instances required more than one minute, 14 required more than five minutes, and only three exceeded ten minutes. The longest run was STRESS_DEMANDPRESSURE_01 at 1,279.001 seconds (21.32 minutes), followed by SCALE_L_24 at 1,176.581 seconds and SENS_MACHINES_HIGH_01 at 890.793 seconds. Thus, every completed case remains well below the one-hour overall limit.

The scale-benchmark medians reveal a pronounced but orderly growth pattern. Small benchmarks require a median of 0.609 seconds, Medium benchmarks 6.224 seconds, and Large benchmarks 129.249 seconds. The corresponding bootstrap intervals for the median are 0.364–1.576 seconds, 2.787–24.446 seconds, and 54.042–286.249 seconds. Sensitivity and stress experiments display greater dispersion because those designs deliberately alter resource tightness and structural balance. Medium sensitivity cases had a median runtime of 32.729 seconds, Large sensitivity cases 302.812 seconds, and operational-stress cases 109.652 seconds. Figure 4 retains the complete empirical distribution rather than reducing each group to a mean, making both central tendency and heterogeneity visible.

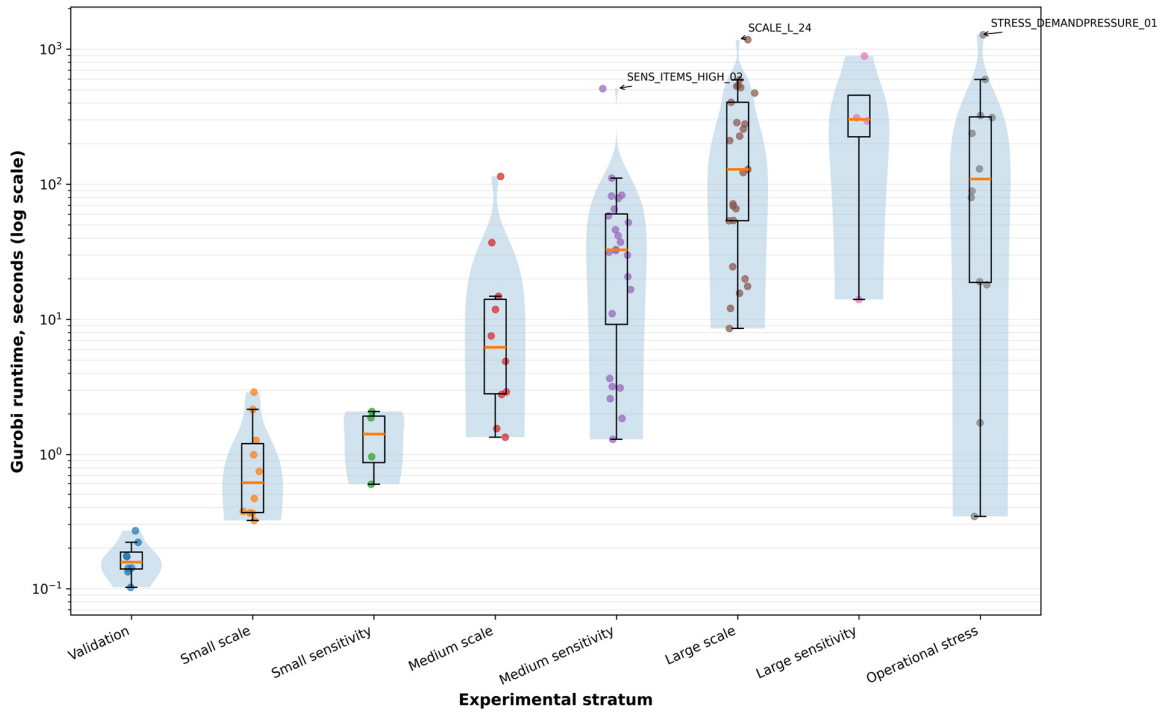

Figure 4. Runtime raincloud for the eight experimental strata. Each distribution combines a density envelope, interquartile box, median, and all individual observations on a logarithmic time scale. The extreme observations are genuine computational outcomes rather than omitted outliers.

4.4.4. Size-adjusted difficulty and hard-instance diagnostics

The residual analysis provides a sharper characterization of computational difficulty than raw runtime alone. SENS_ITEMS_HIGH_02 required 7.79 times the runtime predicted from its sparse variable count, while STRESS_TOOLINGSCARCITY_02 required 7.62 times the size-predicted value. SENS_MACHINES_LOW_03, STRESS_DEMANDPRESSURE_01, SCALE_L_24, and SENS_MACHINES_HIGH_01 required 4.49, 4.19, 3.18, and 3.04 times their respective size-based expectations. These cases represent different mechanisms of difficulty. High item cardinality enlarges the competing sequence and demand-allocation choices; machine scarcity concentrates items on fewer resources; tooling scarcity sharply limits feasible mold routes; and demand pressure makes capacity, outsourcing, and service decisions more tightly coupled.

The tooling-scarcity case is particularly instructive. STRESS_TOOLINGSCARCITY_02 contains only 60,225 variables and solves in 19.036 seconds, so it is not large in absolute terms. Nevertheless, it requires 133 branch-and-bound nodes and is the second most difficult instance after adjusting for size. Conversely, several models containing hundreds of thousands of variables solve at or near the root node and are materially faster than the size trend predicts. The evidence therefore supports a two-dimensional view of scalability: structural scale determines the broad runtime envelope, whereas resource tightness and combinatorial balance determine an instance's position within that envelope.

4.4.5. Incumbent discovery, improvement, and certification

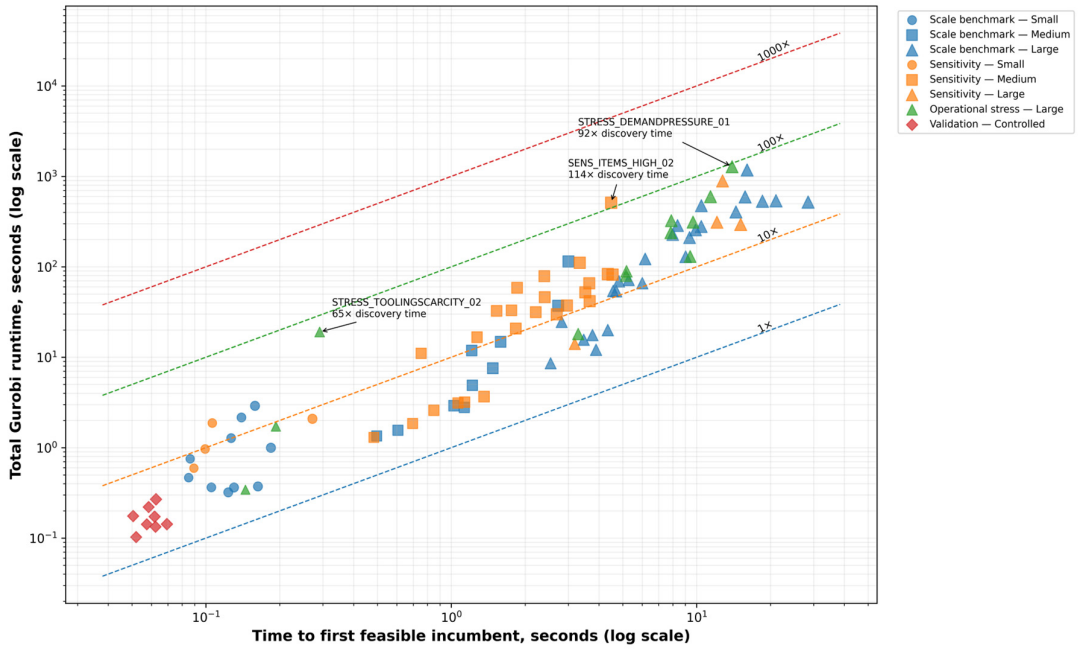

Figure 5. Incumbent discovery-certification phase portrait. Dashed diagonals represent total runtime equal to 1, 10, 100, and 1,000 times the time required to obtain the first feasible incumbent. Distance above the 1x diagonal quantifies post-feasibility search effort.

Feasible schedules are obtained early. The median time to the first incumbent is 2.541 seconds, equal to only 8.46% of total runtime. The median ratio of total runtime to first-incumbent time is 11.83. For the most demanding cases, the difference is much larger: SENS_ITEMS_HIGH_02 runs for approximately 114 times its first-incumbent time, STRESS_DEMANDPRESSURE_01 for 92 times, SCALE_L_24 for 73 times, and SENS_MACHINES_HIGH_01 for 70 times. Figure 5 therefore separates two computational capabilities that should not be conflated: rapid production of a feasible industrial schedule and subsequent effort to improve or certify it.

The phase decomposition in Figure 6 yields an especially clear interpretation. Across the 97 instances, pre-incumbent search occupies a median of 8.46% of runtime, search between the first incumbent and the last rounded incumbent improvement occupies 91.31%, and the post-improvement certification interval occupies only 0.292%. The long-running cases cluster close to the incumbent-improvement vertex. Thus, they are not difficult because feasibility is elusive; they are difficult because numerous near-competitive production, tooling, inventory, and outsourcing combinations remain after a feasible schedule has been identified. This distinction is important for deployment: A planner can often obtain a usable schedule quickly, while additional time predominantly purchases incremental objective improvement and stronger optimality evidence.

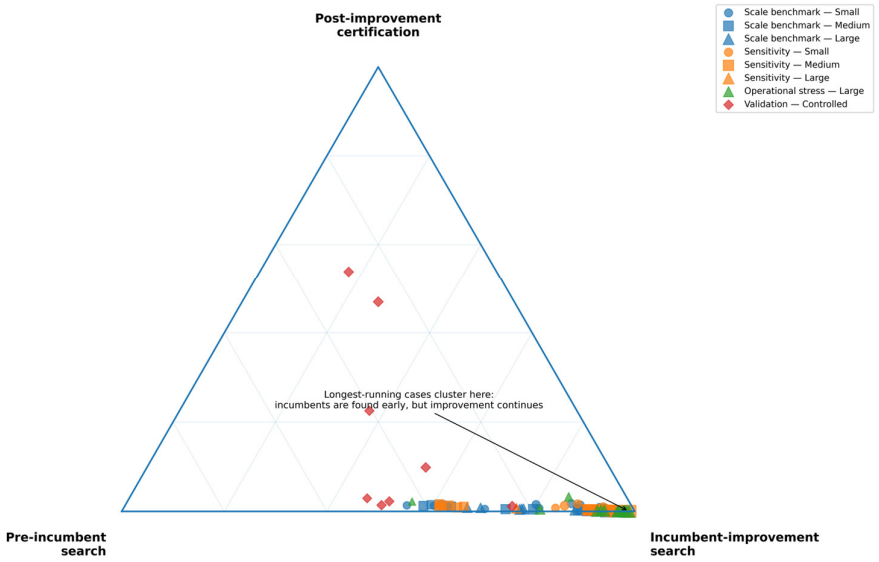

Figure 6. Runtime anatomy simplex. Each point decomposes one solve into pre-incumbent search, incumbent-improvement search, and post-improvement certification. Bubble area increases with total runtime. The concentration near the lower-right vertex shows that incumbent improvement, rather than feasibility discovery, dominates the long solves.

4.4.6. Search-tree behavior, solution quality, and practical tractability

The branch-and-bound evidence is consistent with a comparatively strong formulation. Thirty-nine instances terminate with one processed node, and only five exceed 50 nodes. The maximum is 245 nodes for SENS_ITEMS_HIGH_02; the other cases above 50 nodes are SCALE_L_20, STRESS_DEMANDPRESSURE_01, STRESS_OPERATIONALDISRUPTION_01, and

STRESS_TOOLINGSCARCITY_02. The median number of recorded incumbent solutions is four, with a range of two to ten. These results indicate that Gurobi presolve, root relaxation, and cutting are effective for much of the test set, while a limited subset requires genuine branching because resource competition produces several near-equivalent integer schedules.

Solution quality is correspondingly strong. Ninety-six of 97 runs, or 98.97%, terminated with the Gurobi status OPTIMAL under the specified MIP-gap tolerance. The only interrupted run has a final gap of 0.00450084%, and all 97 solutions pass the independent feasibility and objective-reconstruction checks. The maximum observed runtime is 21.32 minutes, despite models containing up to 1.615 million sparse variables, approximately 576,000 constraints, and 5.640 million nonzero coefficients. The evidence therefore supports practical tractability for the studied industrial range, subject to an important qualification: the runtime envelope is broad, and resource-tight instances can require several times the effort predicted by model size. For operational use, the early-incumbent behavior provides a natural anytime capability, while the rounded-improvement stopping rule offers a defensible mechanism for limiting diminishing-return search on the hardest cases.

4.5. Economic and operational performance

The operational evaluation uses the same 97 completed and independently validated instances described in Sections 4.2–4.4. The aggregate experiment exposure comprises 8,172,672 units of modeled demand. Because each instance represents a separate planning problem, cumulative quantities are used only to characterize the experimental evidence base; they should not be interpreted as a single factory-year production total. All 97 schedules satisfy the independent feasibility and objective-reconstruction checks. Table 6 summarizes the average operational performance across the 25 Large-scale instances, highlighting the improvements achieved by the optimization-based production plan over the existing manual plan in terms of cost, setup time, changeovers, mold utilization, subcontracting, inventory, and on-time completion.

Table 6. Average operational performance of the manual and optimization-based production plans across the 25 Large-scale instances.

KPI	Manual plan	Optimized plan	Improvement
Scaled total cost	264,879	240,842	9.07% reduction
Setup time (h)	615.57	555.43	9.77% reduction
Number of changeovers	784	707.6	9.74% reduction
Mold utilization (%)	31.64%	35.67%	12.74% increase
Subcontracted quantity	895	815	8.94% reduction
Backlog	0.00	0.00	No change
Average ending inventory per item-period (units)	2.63	2.41	8.37% reduction
On-time completion (%)	90.25%	100.00%	10.80% increase

Comparison with the existing manual production plan further demonstrates the operational benefits of the optimization-based scheduling approach. For confidentiality, the monetary cost values were transformed using a common scaling factor; consequently, the reported cost values do not disclose the company's actual monetary figures, while relative cost comparisons and percentage improvements remain unchanged. Across the 25 Large-scale instances, the average scaled total cost

decreased from 264,879 under the manual plan to 240,842 under the optimized plan, corresponding to a 9.07% reduction. Average setup time decreased from 615.57 h to 555.43 h (9.77%), while the average number of changeovers decreased from 784 to 707.6 (9.74%). Average mold utilization increased from 31.64% to 35.67%, representing an improvement of 4.03 percentage points, indicating more effective utilization of shared tooling resources. The average subcontracted quantity decreased from 895 to 815 units (8.94%). Average ending inventory per item-period decreased from 2.63 to 2.41 units (8.37%); this KPI represents the mean inventory carried across item-period combinations rather than the aggregate plant inventory. Backlog remained zero under both planning approaches. Importantly, on-time completion increased from 90.25% under the manual plan to 100% under the optimized schedule, an improvement of 9.75 percentage points. Overall, the results demonstrate simultaneous improvements in economic performance, setup efficiency, tooling utilization, inventory control, subcontracting dependence, and customer-service performance without introducing backlog.

4.5.1. Cost architecture and service preservation

In-house production is the economic core of the solution architecture. When cost components are aggregated across all experiments, internal production accounts for 92.097% of total objective value. Direct subcontracting accounts for 3.495%, and the penalty associated with outsourcing items preferred for internal manufacture contributes a further 4.300%. Inventory carrying contributes only 0.109%. No backlog or grouped-demand delay penalty was incurred in any completed solution. The cost profile is therefore sharply concentrated: The optimizer uses internal production for the overwhelming majority of demand and retains outsourcing as a small but deliberately expensive recourse mechanism.

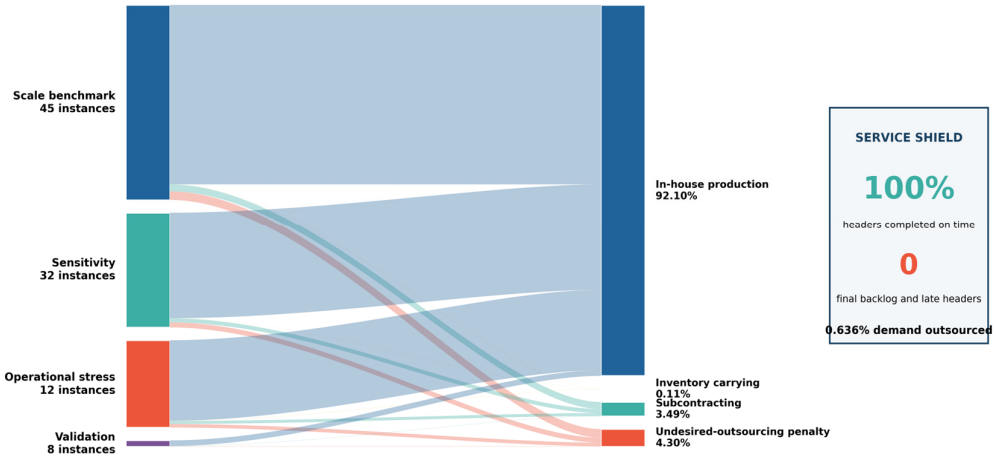

Figure 7. Cost architecture and service preservation across 97 validated experiments. Ribbon width is proportional to aggregate objective contribution. The service shield summarizes the invariant fulfillment outcomes: 100% on-time header completion, zero backlog, and only 0.636% of modeled demand subcontracted.

This distinction between outsourced volume and outsourced cost is operationally important. Only 51,984 units are subcontracted across 8.173 million modeled demand units, corresponding to 0.636%

of total demand. Nevertheless, direct outsourcing and its strategic penalty jointly account for 7.794% of aggregate cost. The model therefore does not use subcontracting as a routine capacity substitute; it uses a small amount at a high marginal cost to protect feasibility and service when machine, mold, setup, maintenance, holiday, or stoppage restrictions make internal timing unattractive. This is precisely the behavior intended by the sourcing-preference penalty.

Across the 97 optimized computational runs, every grouped header is completed on time, quantity-weighted on-time completion is 100%, final and cumulative backlog are zero, and no completion-delay penalty is generated. These results do not imply that service is costless. Rather, they show that the integrated recourse structure, advance production, inventory, alternative molds and machines, and limited outsourcing, absorbs operational pressure before it reaches the customer-facing completion measures. Figure 7 visualizes this cost-to-service architecture.

4.5.2. Setup burden, changeovers, and mold utilization

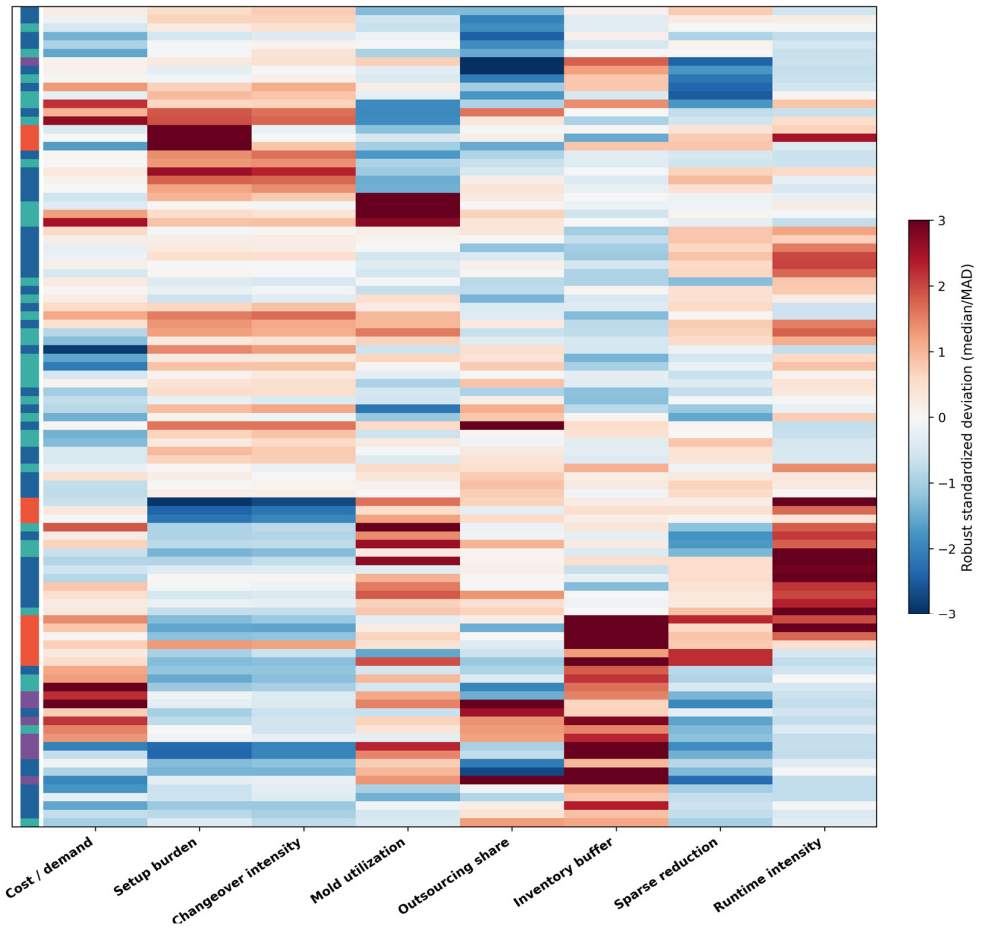

Figure 8. Clustered operational fingerprints of all 97 experiments. Rows are ordered by Ward hierarchical clustering of eight robustly standardized metrics. The narrow strip at the left identifies experiment block: blue, scale benchmark; teal, sensitivity; red, operational stress; purple, controlled validation.

The median schedule incurs 4.206 setup hours and 5.329 effective changeovers per 1,000 demand units. Weighted mold utilization has a median of 35.345%, with a broad range from 17.775% to 77.546%. This dispersion is not evidence of inconsistent planning quality. Low utilization can represent deliberate redundancy, long maintenance exposure, or a large alternative-mold pool, whereas high utilization can indicate tooling concentration or scarcity. Consequently, mold utilization must be interpreted jointly with setup burden, compatibility structure, and service recourse rather than as a stand-alone maximization objective.

The pooled evidence supports this integrated interpretation. Setup hours per 1,000 demand units are negatively associated with weighted mold utilization (Spearman $\rho = -0.363$, bootstrap 95% interval -0.541 to -0.162). High setup burden consumes slot time that would otherwise be productive and it fragments the periods in which assigned molds generate output. This association is descriptive rather than causal because model dimensions and scenario design vary across the experiment set, but its direction is consistent with the capacity equation in which setup time is deducted from the productive slot length.

Figure 8 shows that operational difficulty is multidimensional. Hierarchical clustering does not arrange instances into a simple Small–Medium–Large progression. Instead, clusters emerge from combinations of setup burden, changeover intensity, tooling utilization, outsourcing, inventory, sparsity, and runtime intensity. High-setup stress cases form a distinctive setup-dominated signature; operational-disruption cases are distinguished by inventory buffering; low-mold sensitivity cases exhibit high tooling utilization; and controlled validation cases remain compact computationally while deliberately activating selected physical mechanisms.

4.5.3. Inventory buffering and the hierarchy of recourse

Inventory is inexpensive in objective terms but strategically important in feasibility terms. Its aggregate cost share is only 0.109%, yet disruption-oriented instances use it strongly. Across all 97 cases, average inventory has positive rank associations with the number of stoppage events ($\rho = 0.287$; bootstrap 95% interval 0.164–0.427) and holiday periods ($\rho = 0.383$; bootstrap 95% interval 0.175–0.527). These relationships indicate that the model anticipates inaccessible production time by shifting production earlier and carrying stock across periods. Because the model simultaneously preserves minimum ending inventory and grouped-demand completion, inventory acts as a temporal bridge rather than as uncontrolled overproduction.

Taken together, the results reveal a recourse hierarchy. The optimizer first exploits feasible machine–mold alternatives and sequence consolidation; it then uses advance production and inventory to bridge future capacity losses; and it resorts to penalized subcontracting only for a very small residual volume. Backlog and customer delay are the last recourse options and are not used in the completed experiment set. The hierarchy explains how service can remain perfect even when setup, tooling, maintenance, and demand conditions differ substantially.

4.6. Sensitivity analysis

4.6.1. Analytical design and interpretation

The sensitivity program contains four designed factors, machines, items, molds, and planning periods, each evaluated at Low and High levels using four instances from the case-study company per

level. Because only two levels are available and some non-target dimensions vary within their prescribed production-line or full-line ranges, the analysis is presented as a designed-level contrast rather than as a continuous response surface or universal causal elasticity. For each response, Table 7 reports the Low- and High-level medians, their ratio, Cliff's delta as a distribution-free effect size, and the exact two-sided Mann–Whitney probability. With four observations per level, the smallest attainable two-sided probability was 0.0286; therefore, effect magnitude and operational coherence are more informative than binary significance labels.

Table 7. Sensitivity analysis: designed-level effects across machines, items, molds, and planning periods.

Factor	Response	Low median	High median	High/Low	Cliff's δ	Exact p
Machines	Sparse variables	19,505	1,112,616	57.043	1.000	0.0286
Machines	Solver runtime	1.419	302.812	213.376	1.000	0.0286
Machines	Cost per demand unit	2.056	1.803	0.877	-0.750	0.1143
Machines	Setup hours / 1,000 units	3.635	4.438	1.221	0.750	0.1143
Machines	Changeovers / 1,000 units	4.432	5.850	1.320	0.875	0.0571
Machines	Weighted mold utilization	34.949	41.331	1.183	0.500	0.3429
Machines	Subcontracted demand	0.684	0.622	0.910	-0.125	0.8857
Machines	Average inventory	3.683	2.388	0.648	-1.000	0.0286
Items	Sparse variables	132,647	377,681	2.847	1.000	0.0286
Items	Solver runtime	3.154	53.694	17.026	1.000	0.0286
Items	Cost per demand unit	1.976	1.851	0.937	-0.750	0.1143
Items	Setup hours / 1,000 units	4.951	4.003	0.808	-0.750	0.1143
Items	Changeovers / 1,000 units	6.395	5.084	0.795	-0.875	0.0571
Items	Weighted mold utilization	29.896	35.336	1.182	0.250	0.6857
Items	Subcontracted demand	0.567	0.646	1.139	0.625	0.2000
Items	Average inventory	2.802	2.419	0.863	-0.250	0.6857
Molds	Sparse variables	222,594	261,070	1.173	0.125	0.8857
Molds	Solver runtime	24.074	33.439	1.389	0.125	0.8857
Molds	Cost per demand unit	2.012	1.895	0.942	-0.375	0.4857
Molds	Setup hours / 1,000 units	4.437	4.450	1.003	0.125	0.8857
Molds	Changeovers / 1,000 units	5.525	5.822	1.054	0.500	0.3429
Molds	Weighted mold utilization	58.401	27.539	0.472	-1.000	0.0286
Molds	Subcontracted demand	0.693	0.588	0.848	-0.500	0.3429
Molds	Average inventory	2.601	2.442	0.939	-0.375	0.4857
Periods	Sparse variables	216,582	380,884	1.759	0.750	0.1143
Periods	Solver runtime	34.808	57.240	1.644	0.500	0.3429
Periods	Cost per demand unit	1.887	1.794	0.950	-0.500	0.3429

Continued on next page

Factor	Response	Low median	High median	High/Low	Cliff's δ	Exact p
Periods	Setup hours / 1,000 units	4.488	4.228	0.942	-0.125	0.8857
Periods	Changeovers / 1,000 units	5.671	5.276	0.930	-0.125	0.8857
Periods	Weighted mold utilization	36.588	35.402	0.968	0.000	1.0000
Periods	Subcontracted demand	0.637	0.670	1.052	0.375	0.4857
Periods	Average inventory	2.920	1.917	0.657	-1.000	0.0286

4.6.2. Machine-level contrast

The High-machine level is a composite large-scale contrast rather than a pure marginal addition of presses, because the associated item, mold, and horizon ranges are also larger. Its median sparse model contains 57.04 times as many variables and requires 213.38 times the runtime of the Low-machine level; both groups are completely separated (Cliff's $\delta = 1.00$, exact $p = 0.0286$). Operationally, the High level has 12.3% lower median cost per demand unit and 35.2% lower average inventory, while changeovers per 1,000 demand units are 32.0% higher. The contrast demonstrates that larger resource systems can exploit scale and reduce buffering, but their enlarged sequencing space dominates computation. It should not be interpreted as evidence that adding machines alone multiplies runtime by 213.

4.6.3. Item-cardinality contrast

Doubling the designed item level from 20 to 40 has a disproportionate computational effect. Median sparse variables rise by a factor of 2.85, whereas runtime rises by a factor of 17.03 ($\delta = 1.00$, $p = 0.0286$). This confirms that item cardinality expands more than production-quantity rows: It enlarges machine-specific transition arcs, setup combinations, item–mold alternatives, and grouped-demand allocations. In contrast, normalized operating effort decreases: Setup hours and changeovers per 1,000 demand units fall to 0.81 and 0.79 of the Low-level medians. The most plausible interpretation is campaign-scale dilution; larger product portfolios in these case-study instances are accompanied by larger aggregate demand, so a changeover supports more output even though the combinatorial search becomes more difficult.

4.6.4. Mold-cardinality contrast

The mold contrast provides the clearest tooling-management result. Increasing the mold level from 15 to 30 changes median sparse variables by only 17.3% and median runtime by 38.9%, but weighted mold utilization falls from 58.40% to 27.54%. The distributions were separated ($\delta = -1.00$, $p = 0.0286$). Furthermore, median subcontracted demand falls by approximately 15.2%, and cost per demand unit falls by 5.8%. Additional molds therefore buy routing flexibility and reduce dependence on external supply, but the price is lower individual-tool utilization. This is not inefficiency in the conventional sense; it is the measurable cost of resilience and alternative capacity.

4.6.5. Planning-horizon contrast

Increasing the horizon from seven to ten periods raises median sparse variables by 75.9% and runtime by 64.4%. The growth is material but less explosive than the item-cardinality effect because added periods replicate temporal decision structures without generating the same quadratic increase in sequence alternatives. Moreover, average inventory falls to 65.7% of the Low-period median ($\delta = -1.00$, $p = 0.0286$), while cost per demand unit decreases by approximately 5.0%. The additional temporal flexibility enables production and demand allocation to be distributed more selectively across the horizon, reducing the average stock carried at an item–period level. Table 7 summarizes all four designed-level contrasts without suppressing small-sample uncertainty.

4.7. Operational stress analysis

4.7.1. Matched-comparison protocol

The 12 stress experiments contain four scenarios with three instances from the case-study company each: high setup intensity, tooling scarcity, operational disruption, and demand pressure. Raw comparisons are potentially misleading because the stress instances differ in dimensions and demand. Each stress case was therefore compared with the median of its three nearest Large scale benchmarks in standardized machine, item, mold, period, slot, and total-demand space. The reported stress ratios are medians across three replications. This nearest-neighbor procedure improves structural comparability, but it remains an observational normalization rather than a randomized causal estimator.

4.7.2. High setup intensity

High setup intensity produces the strongest direct capacity shock. Setup hours per 1,000 demand units are 2.88 times the matched-benchmark level, while runtime per 100,000 variables is 1.32 times higher and mold utilization falls to 0.87 of the matched level; yet, cost per demand unit remains at 0.97 of the benchmark and subcontracting at 1.03. The schedule therefore absorbs the setup shock primarily through sequencing and capacity reallocation rather than through a large increase in external supply or unit cost. This result demonstrates the value of embedding sequence-dependent setup time directly in productive-capacity constraints: The operational consequence is visible without destabilizing service.

4.7.3. Tooling scarcity

Tooling scarcity creates a counterintuitive computational pattern. Average inventory is 2.09 times the matched-benchmark level, indicating reduced routing flexibility and greater reliance on temporal buffering. However, runtime per 100,000 variables is slightly lower, at 0.95 of the matched level. The reason is structural: the tooling-scarcity instances have machine–item densities around 9% and feasible machine–item–mold triple densities near 0.20–0.30%, so the model contains far fewer assignment and transition alternatives. Scarcity can therefore make the planning problem operationally less flexible while making the mathematical search space smaller. Computational ease must not be mistaken for operational abundance.

4.7.4. Operational disruption

Operational disruption is absorbed principally through inventory. Relative to matched Large benchmarks, average inventory increases by a factor of 13.63, weighted mold utilization by 1.13, and runtime intensity by 1.28, whereas cost per demand unit increases by only 1.4%. The three disruption cases have average inventory between 25.32 and 31.70 units, far above the overall median of 2.59. This is the clearest evidence that the model anticipates holidays, maintenance, and stoppages rather than reacting after demand becomes late. Inventory functions as a planned protection mechanism, not as an accidental residual.

4.7.5. Demand pressure

Demand pressure raises median mold utilization to 1.19 times the matched-benchmark level and average inventory to 1.08, while cost per demand unit remains effectively unchanged at 1.00 and runtime intensity at 1.06. Setup hours and changeovers per 1,000 units fall to 0.56 and 0.57 of matched values. The decline in normalized switching effort is consistent with longer production campaigns under higher throughput. The stress response therefore resembles an economy-of-scale mechanism: tooling is used more intensively and each setup supports a larger volume, while the integrated plan preserves both cost stability and service.

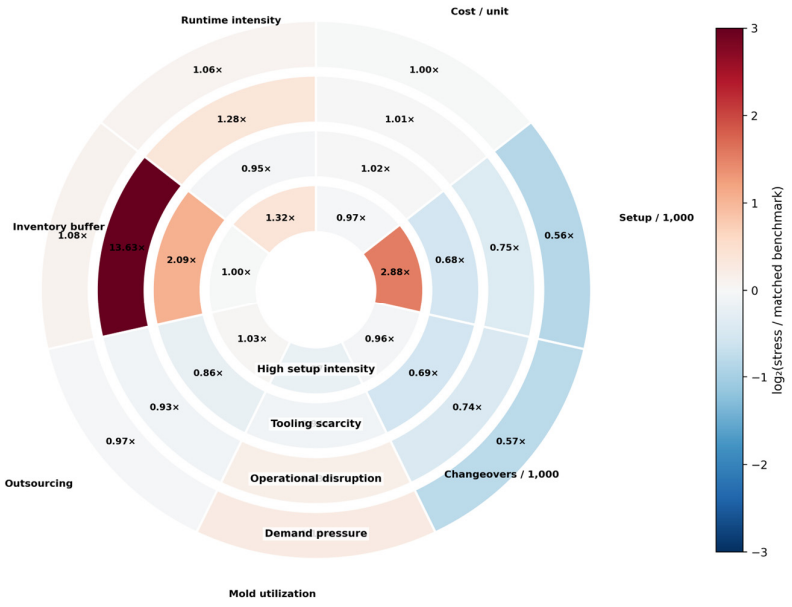

Figure 9. Stress-propagation wheel relative to dimension-matched Large benchmarks. Each cell reports the median stress-to-benchmark ratio across three replications. Red indicates a response above the matched baseline and blue a response below it.

Figure 9 shows how each operational stress scenario propagates through cost, setup burden, changeover intensity, mold utilization, subcontracting, inventory, and size-normalized runtime relative to dimension-matched Large benchmarks. The wheel makes the different response signatures visible:

High setup intensity is dominated by setup burden, tooling scarcity raises inventory while reducing alternative routes, operational disruption produces the strongest inventory-buffer response, and demand pressure increases mold utilization while lowering switching effort per unit.

4.7.6. Operational structure of a disrupted large instance

Figure 10 shows the detailed production schedule for STRESS_OPERATIONALDISRUPTION_01, a Large case-study instance with 22 active machines, 79 items, 49 molds, eight periods, seven exact stoppages, one holiday period, and eight active mold-maintenance events. Nine representative machines are displayed to preserve the visibility of item–mold assignments, setup portions, and disruption intervals. The Gantt chart demonstrates three features that aggregate KPIs cannot show. First, stoppages are embedded as exact zero-production intervals rather than approximated as capacity reductions. Second, item–mold combinations change across slots while physical mold exclusivity is preserved. Third, production is redistributed around the holiday, maintenance, and stoppage windows without producing backlog or late header completion.

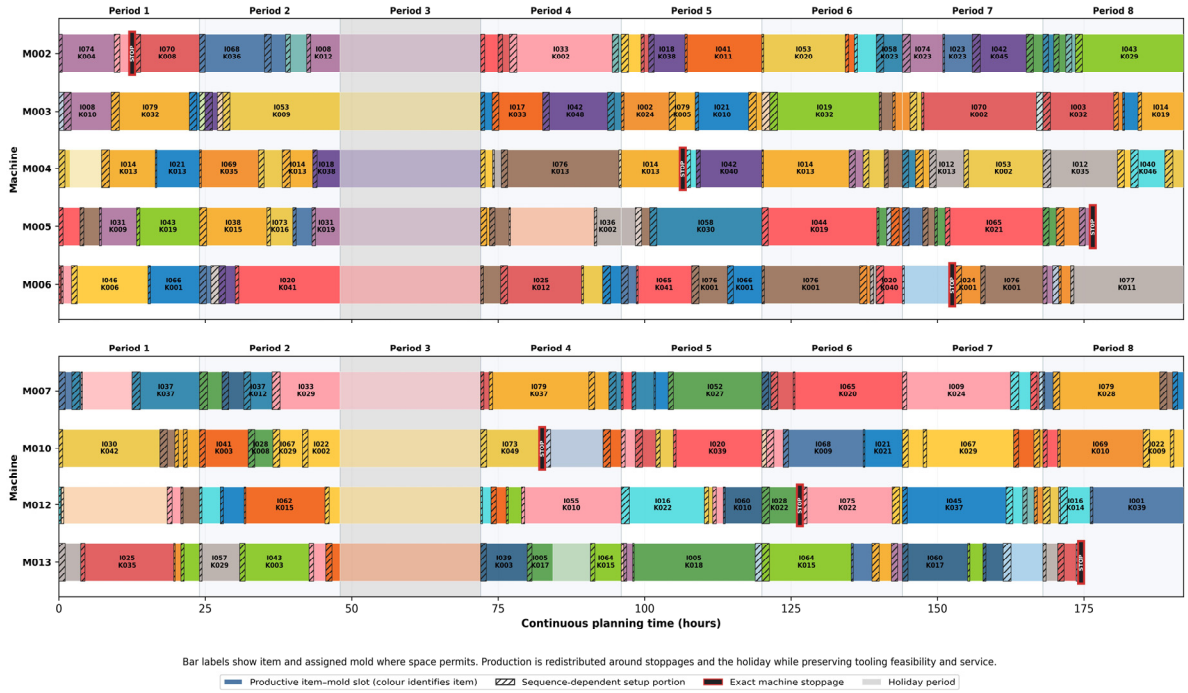

Figure 10. Representative Large-instance production-schedule Gantt chart for STRESS_OPERATIONALDISRUPTION_01. The chart shows nine representative machines from the 22 active machines in the instance. Colours identify scheduled items, bar labels identify the item and assigned mold where space permits, hatched leading portions show sequence-dependent setup time, black bars with red outlines show exact stoppages, and the grey band denotes the holiday period.

4.8. Mathematical validation and solution credibility

The computational claims are supported by eight controlled validation cases: two independent replications each for timing and shift boundaries, shared-mold exclusivity, maintenance and holiday alternatives, and grouped-demand completion with backlog. Every case has an incumbent, passes the complete independent verification procedure for constraints (1)–(54), contains no sparse-domain violation, passes the objective coefficient audit, reproduces the Gurobi objective independently, and matches its declared experiment identity and dimensions.

The resulting schedules provide additional mechanism-specific evidence. All eight cases have zero period-bound violations. Shared-mold validation cases have a maximum of one machine assigned to any mold–period–slot combination. The two maintenance/holiday cases contain 16 and 12 active mold-maintenance restrictions and 20 holiday machine-slot restrictions each; every holiday-restricted slot has zero production and zero setup. Both grouped-demand cases complete all 20 demand headers on time and have zero ending backlog. The timing cases contain the intended shift transition gaps while maintaining zero within-period boundary violation; these gaps are permissible shift-calendar discontinuities rather than slot-continuity failures.

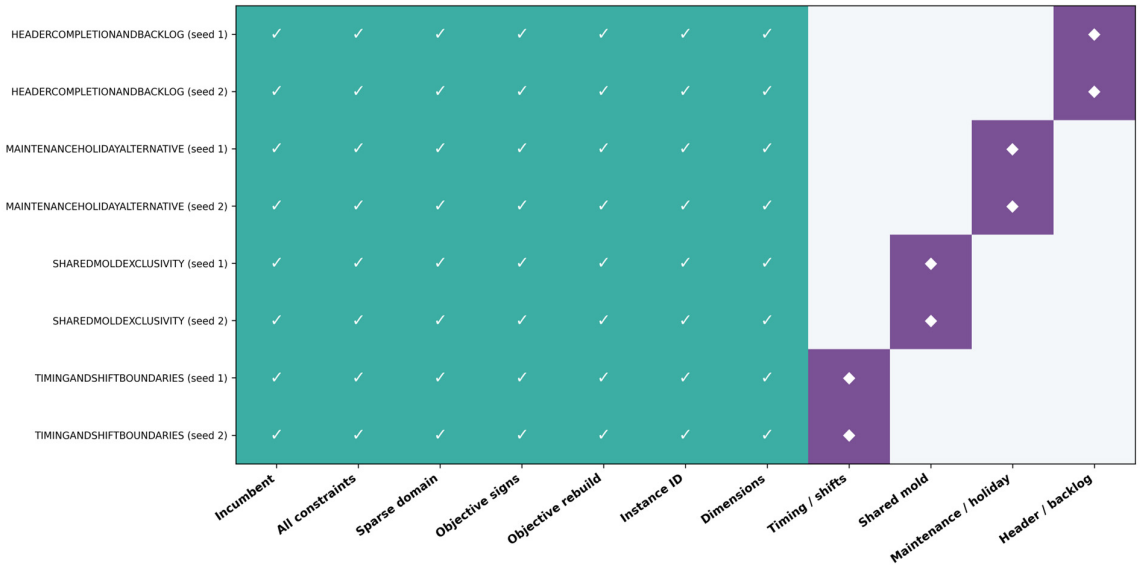

Figure 11. Controlled validation evidence lattice. Check marks denote general properties independently verified in every case; diamonds identify the mechanism deliberately targeted by each controlled scenario. Both replications reproduce the intended evidence pattern.

Figure 11 separates general verification from scenario-specific targeting. The matrix shows that passing a generic solver feasibility check is necessary but not sufficient. Credibility is stronger when a controlled case activates the precise mechanism under examination and the independently verified solution exhibits the required physical behavior.

4.9. Integrated managerial insights

The complete experiment program supports five integrated managerial conclusions. First, service reliability is achieved through a hierarchy of internal flexibility, inventory, and selective outsourcing rather than through backlog. The 100% service result is therefore meaningful only when interpreted together with the small outsourced volume and its disproportionately high cost share. Second, secondary tooling should be managed as a resilience resource, not solely as a utilization target. Additional molds reduce utilization but also reduce outsourcing exposure; a lower utilization percentage may be the rational consequence of deliberate redundancy.

Third, setup management has operational and computational value. High setup intensity consumes nearly three times the normalized setup capacity of matched benchmarks and raises size-normalized runtime. Investment in family sequencing, mold-change standardization, and improved setup-time data can therefore improve both schedule economics and optimization performance. Fourth, inventory is the dominant protection against planned disruption. The large inventory response under holidays, stoppages, and maintenance is not evidence of poor control; it is the model's least-cost method of preserving customer commitments when future production windows are known to be inaccessible.

Fifth, the resource that is most valuable depends on the operating regime. Additional molds create routing resilience and lower outsourcing but dilute utilization. Additional temporal horizon reduces average inventory and cost per unit while enlarging the model. Higher item cardinality creates a far stronger computational burden than its variable-count increase alone would suggest. Tooling scarcity can be computationally easy because it removes alternatives, even though it is operationally restrictive. These distinctions demonstrate why machine count, mold count, utilization, runtime, and service should not be interpreted independently. The value of the formulation lies precisely in exposing their joint trade-offs within a single schedule.

5. Conclusions

In this study, we developed and computationally evaluated an integrated mixed-integer programming formulation for injection-molding production planning in which machine capacity, physical mold availability, sequence-dependent setups, inventory, subcontracting, planned stoppages, holidays, mold maintenance, and grouped customer demand are represented within one scheduling model. The central modeling premise is that feasible capacity is jointly created by machines and secondary shared tooling. Treating molds only as item attributes or preprocessing filters would therefore omit a resource interaction that directly determines sequence feasibility, productive time, and service recovery.

The first contribution is the integrated representation. The formulation combines continuous-time slot scheduling with explicit setup states, machine–item eligibility, mold–item and mold–machine compatibility, physical mold exclusivity, exact disruption intervals, multi-period inventory and backlog conservation, selective subcontracting, and grouped-demand completion. Sparse variable construction on verified eligibility sets preserves the feasible production decisions while avoiding dense Cartesian domains that have no physical interpretation. Across the 97 completed experiments, sparse construction reduced the theoretical variable count by a median of 83.00%, with reductions of 84.78%–89.79% throughout the ordinary Large benchmark class.

The second contribution is the scale and rigor of the computational evidence. The computational program contains 45 scale benchmarks, 32 designed sensitivity experiments, 12 operational-stress experiments, and eight controlled validation cases. Ninety-six runs terminated with the Gurobi status OPTIMAL under the specified MIP-gap tolerance, while the only interrupted run had a final relative gap of 0.00450084% and was not relabeled as optimal. Every completed run passed the independent feasibility, sparse-domain, objective-reconstruction, identity, and KPI checks. The largest formulation contained 1,614,714 sparse variables and 5,640,496 nonzero coefficients, and the largest constraint system contained 576,357 constraints. Median solver time was 20.749 s and the maximum was 1,279.001 s (21.32 min) on the stated workstation. Feasible incumbents were generally available much earlier than final termination, indicating a useful anytime property for operational deployment.

The third contribution is the joint economic and operational interpretation. Across 8,172,672 modeled demand units, every grouped demand header was completed on time, final and cumulative backlog were zero, and only 0.636% of demand was subcontracted. That small outsourced volume nevertheless represented 7.794% of aggregate objective value when direct subcontracting and the sourcing-preference penalty were combined. The schedules therefore reveal a consistent recourse hierarchy: Machine–mold flexibility and sequence consolidation are used first, advance production and inventory bridge known capacity losses, and penalized subcontracting protects residual service requirements. Backlog and customer delay remain last-resort mechanisms and were not used in the complete experiment set.

The sensitivity and stress analyses further show that operational value cannot be inferred from a single utilization or size measure. Additional molds reduce individual-tool utilization but improve routing resilience and reduce outsourcing exposure. Greater item cardinality expands the transition and allocation space much faster than its variable-count increase alone suggests. Planned disruption is absorbed primarily through inventory, whereas demand pressure produces longer campaigns and lower setup effort per unit. Tooling scarcity can make the plant less flexible while making the mathematical problem smaller by eliminating alternatives. These findings support integrated interpretation of machines, molds, setup burden, inventory, outsourcing, service, and runtime rather than isolated KPI maximization.

The present scalability conclusions are restricted to the industrial range evaluated in this study, whose full case-study plant contains 24 injection-molding machines. Accordingly, the computational results should not be extrapolated directly to substantially larger facilities with 30–50 or more machines. Because the number of sequencing and machine–item–mold alternatives can grow superlinearly with system size, such environments may require additional computational strategies. Promising extensions include rolling-horizon decomposition, machine or time-based decomposition, MIP warm starts, matheuristic approaches, and hybrid exact–heuristic methods that retain the integrated resource logic while reducing the size of each optimization subproblem.

Data Availability

The data used in this study were obtained from an industrial organization and contain commercially sensitive and confidential information. Therefore, the data are not publicly available. Access may be considered upon reasonable request to the corresponding authors, subject to approval from the organization that provided the data.

Author contributions

M. Sankar contributed to the conceptualization, methodology, investigation, software development, data curation, validation, and writing of the original draft. M. Ramakrishnan contributed to supervision, and the review and editing of the manuscript. All authors have read and agreed to the published version of the manuscript.

Use of Generative-AI tools declaration

During the preparation of this manuscript, the authors used Gemini solely for the purpose of language editing, grammatical correction, and improving the overall readability of the text. Following the use of this tool, the authors manually reviewed and edited the content as necessary and take full responsibility for the final integrity and accuracy of the work.

Conflict of interest

The authors declare no conflict of interest.

References

1. R. Geyer, J. R. Jambeck, K. L. Law, Production, use, and fate of all plastics ever made, *Sci. Adv.*, **3** (2017), e1700782. https://doi.org/10.1126/sciadv.1700782
2. A. L. Andrady, M. A. Neal, Applications and societal benefits of plastics, *Philos. Trans. R. Soc. B Biol. Sci.*, **364** (2009), 1977–1984. https://doi.org/10.1098/rstb.2008.0304
3. J. Hopewell, R. Dvorak, E. Kosior, Plastics recycling: challenges and opportunities, *Philos. Trans. R. Soc. B Biol. Sci.*, **364** (2009), 2115–2126. https://doi.org/10.1098/rstb.2008.0311
4. D. V. Rosato, D. V. Rosato, M. G. Rosato, *The Complete Injection Molding Process,* Boston: Springer, 2000. https://doi.org/10.1007/978-1-4615-4597-2
5. Z. Chen, L. S. Turng, A review of current developments in process and quality control for injection molding, *Adv. Polym. Technol.*, **24** (2005), 165–182. https://doi.org/10.1002/adv.20046
6. K. Amellal, C. Tzoganakis, A. Penlidis, G. L. Rempel, Injection molding of medical plastics: a review, *Adv. Polym. Technol.*, **13** (1994), 315–322. https://doi.org/10.1002/adv.1994.060130407
7. A. Thiriez, T. Gutowski, An environmental analysis of injection molding, *Proc. IEEE Int. Symp. Electron. Environ.,* 2006, 195–200. https://doi.org/10.1109/ISEE.2006.1650060
8. E. B. Edis, C. Oguz, I. Ozkarahan, Parallel machine scheduling with additional resources: notation, classification, models and solution methods, *Eur. J. Oper. Res.*, **230** (2013), 449–463. https://doi.org/10.1016/j.ejor.2013.02.042
9. H. Stadtler, Multilevel lot sizing with setup times and multiple constrained resources: internally rolling schedules with lot-sizing windows, *Oper. Res.*, **51** (2003), 487–502. https://doi.org/10.1287/opre.51.3.487.14949
10. A. Allahverdi, J. N. D. Gupta, T. Aldowaisan, A review of scheduling research involving setup considerations, *Omega*, **27** (1999), 219–239. https://doi.org/10.1016/S0305-0483(98)00042-5
11. A. Allahverdi, C. T. Ng, T. C. E. Cheng, M. Y. Kovalyov, A survey of scheduling problems with setup times or costs, *Eur. J. Oper. Res.*, **187** (2008), 985–1032. https://doi.org/10.1016/j.ejor.2006.06.060

12. A. Allahverdi, The third comprehensive survey on scheduling problems with setup times/costs, *Eur. J. Oper. Res.*, **246** (2015), 345–378. https://doi.org/10.1016/j.ejor.2015.04.004
13. P. Brucker, A. Drexl, R. Möhring, K. Neumann, E. Pesch, Resource-constrained project scheduling: notation, classification, models, and methods, *Eur. J. Oper. Res.*, **112** (1999), 3–41. https://doi.org/10.1016/S0377-2217(98)00204-5
14. B. Karimi, S. M. T. Fatemi Ghomi, J. M. Wilson, The capacitated lot sizing problem: a review of models and algorithms, *Omega*, **31** (2003), 365–378. https://doi.org/10.1016/S0305-0483(03)00059-8
15. A. Drexl, A. Kimms, Lot sizing and scheduling: survey and extensions, *Eur. J. Oper. Res.*, **99** (1997), 221–235. https://doi.org/10.1016/S0377-2217(97)00030-1
16. H. Meyr, Simultaneous lot-sizing and scheduling by combining local search with dual reoptimization, *Eur. J. Oper. Res.*, **120** (2000), 311–326. https://doi.org/10.1016/S0377-2217(99)00159-9
17. H. M. Wagner, T. M. Whitin, Dynamic version of the economic lot size model, *Manag. Sci.*, **5** (1958), 89–96. https://doi.org/10.1287/mnsc.5.1.89
18. A. S. Manne, On the job-shop scheduling problem, *Oper. Res.*, **8** (1960), 219–223. https://doi.org/10.1287/opre.8.2.219
19. Y. Pochet, L. A. Wolsey, *Production Planning by Mixed Integer Programming,* Boston: Springer, 2006. https://doi.org/10.1007/0-387-33477-7
20. M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems,* 4th edition, Boston: Springer, 2012. https://doi.org/10.1007/978-1-4614-2361-4
21. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys, *Handbooks in Operations Research and Management Science*, 1993. https://doi.org/10.1016/S0927-0507(05)80189-6
22. J. M. Framinan, J. N. D. Gupta, R. Leisten, A review and classification of heuristics for permutation flow-shop scheduling with makespan objective, *J. Oper. Res. Soc.*, **55** (2004), 1243–1255. https://doi.org/10.1057/palgrave.jors.2601784
23. J. H. Holland, *Adaptation in Natural and Artificial Systems,* MIT Press, 1992. https://doi.org/10.7551/mitpress/1090.001.0001
24. F. Glover, Tabu search - Part I, *ORSA J. Comput.*, **1** (1989), 190–206. https://doi.org/10.1287/ijoc.1.3.190
25. S. Kirkpatrick, C. D. Gelatt Jr., M. P. Vecchi, Optimization by simulated annealing, *Science*, **220** (1983), 671–680. https://doi.org/10.1126/science.220.4598.671
26. R. Jans, Z. Degraeve, Meta-heuristics for dynamic lot sizing: a review and comparison of solution approaches, *Eur. J. Oper. Res.*, **177** (2007), 1855–1875. https://doi.org/10.1016/j.ejor.2005.12.008
27. C. R. Sox, J. A. Muckstadt, Optimization-based planning for the stochastic lot-scheduling problem, *IIE Trans.*, **29** (1997), 349–357. https://doi.org/10.1023/A:1018543817586
28. H. Tempelmeier, *Inventory Management in Supply Networks: Problems, Models, Solutions,* 2nd edition, Books on Demand, 2011.
29. K. Copil, M. Wörbelauer, H. Meyr, H. Tempelmeier, Simultaneous lotsizing and scheduling problems: a classification and review of models, *OR Spectr.*, **39** (2017), 1–64. https://doi.org/10.1007/s00291-015-0429-4
30. J. N. Hooker, *A hybrid method for planning and scheduling, in Principles and Practice of Constraint Programming-CP 2004,* Berlin: Springer, 2004. https://doi.org/10.1007/978-3-540-30201-8_24
31. K. Kogan, Optimal scheduling of parallel machines with constrained resources, *Eur. J. Oper. Res.*, **170** (2006), 771–787. https://doi.org/10.1016/j.ejor.2004.05.021

32. B. Fleischmann, The discrete lot-sizing and scheduling problem with sequence-dependent setup costs, *Eur. J. Oper. Res.*, **75** (1994), 395–404. https://doi.org/10.1016/0377-2217(94)90083-3
33. B. Fleischmann, H. Meyr, The general lotsizing and scheduling problem, *OR Spectr.*, **19** (1997), 11–21. https://doi.org/10.1007/BF01539800
34. K. Haase, A. Kimms, Lot sizing and scheduling with sequence-dependent setup costs and times and efficient rescheduling opportunities, *Int. J. Prod. Econ.*, **66** (2000), 159–169. https://doi.org/10.1016/S0925-5273(99)00119-X
35. D. Gupta, T. Magnusson, The capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times, *Comput. Oper. Res.*, **32** (2005), 727–747. https://doi.org/10.1016/j.cor.2003.08.014
36. B. Almada-Lobo, D. Klabjan, M. A. Carravilla, J. F. Oliveira, Single machine multi-product capacitated lot sizing with sequence-dependent setups, *Int. J. Prod. Res.*, **45** (2007), 4873–4894. https://doi.org/10.1080/00207540601094465
37. A. R. Clark, S. J. Clark, Rolling-horizon lot-sizing when setup times are sequence-dependent, *Int. J. Prod. Res.*, **38** (2000), 2287–2307. https://doi.org/10.1080/00207540050028106
38. M. Gopalakrishnan, D. M. Miller, C. P. Schmidt, A framework for modelling setup carryover in the capacitated lot sizing problem, *Int. J. Prod. Res.*, **33** (1995), 1973–1988. https://doi.org/10.1080/00207549508904793
39. M. Gopalakrishnan, A modified framework for modelling set-up carryover in the capacitated lotsizing problem, *Int. J. Prod. Res.*, **38** (2000), 3421–3424. https://doi.org/10.1080/002075400418324
40. G. Fandel, C. Stammen-Hegene, Simultaneous lot sizing and scheduling for multi-product multi-level production, *Int. J. Prod. Econ.*, **104** (2006), 308–316. https://doi.org/10.1016/j.ijpe.2005.04.011
41. M. F. Anwar, R. Nagi, Integrated lot-sizing and scheduling for just-in-time production of complex assemblies with finite set-ups, *Int. J. Prod. Res.*, **35** (1997), 1447–1470. https://doi.org/10.1080/002075497195416
42. K. Kogan, Scheduling under common due date, a single resource and precedence constraints - a dynamic approach, *Discrete Event Dyn. Syst.*, **8** (1998), 353–364. https://doi.org/10.1023/A:1008345132480
43. S. G. Dastidar, R. Nagi, Scheduling injection molding operations with multiple resource constraints and sequence dependent setup times and costs, *Comput. Oper. Res.*, **32** (2005), 2987–3005. https://doi.org/10.1016/j.cor.2004.04.012
44. P. Winklehner, V. A. Hauder, Flexible job-shop scheduling with release dates, deadlines and sequence dependent setup times: a real-world case, *Procedia Comput. Sci.*, **200** (2022), 1654–1663. https://doi.org/10.1016/j.procs.2022.01.366
45. N. Troncoso, H. Cancela, P. Piñeyro, F. Quezada, Ó. C. Vásquez, The product-mold-machine manufacturing problem: Complexity, MILP models and constructive heuristics, *Comput. Ind. Eng.*, **189** (2024), 109937. https://doi.org/10.1016/j.cie.2024.109937
46. Y. Bok, N. K. Lee, S. Jo, S. Lee, S. J. Kweon, H. S. Na, The production scheduling problem employing non-identical parallel machines with due dates considering carbon emissions and multiple types of energy sources, *Expert Syst. Appl.*, **238** (2024), 121990. https://doi.org/10.1016/j.eswa.2023.121990
47. N. G. Kim, S. Yoon, S. J. Kweon, H. S. Na, Complex multi-stage production scheduling with unrelated parallel machines under carbon regulation: balancing cost efficiency and carbon emissions, *Comput. Ind. Eng.*, **219** (2026), 112152. https://doi.org/10.1016/j.cie.2026.112152

48. S. Jo, H. S. Na, S. Yoon, S. J. Kweon, An integrated framework for solving the green supplier selection and order allocation problem in steam procurement, *Expert Syst. Appl.*, **312** (2026), 131386. https://doi.org/10.1016/j.eswa.2026.131386
49. M. Capponi, W. Behiri, S. Belmokhtar-Berraf, M. Sali, A robust ant colony algorithm for batching and sequencing problem: A case study in injection molding, *IFAC-PapersOnLine*, **59** (2025), 1689–1694. https://doi.org/10.1016/j.ifacol.2025.09.284
50. A. H. Akbari, M. Jafari, P. Akhavan, A data-driven multi-objective optimization framework for dynamic job shop scheduling with order acceptance, inventory and energy-aware decisions, *Comput. Ind. Eng.*, **214** (2026), 111886. https://doi.org/10.1016/j.cie.2026.111886

AIMS Press