Reactive Scheduling under Due-Date and Intermediate Storage Constraints: A Conceptual Multi-Objective Approach

Krisztián Attila Bakon, Tibor Holczinger
Faculty of Information Technology, University Center for Circular Economy Nagykanizsa, University of Pannonia
Department of Applied Informatics
Zrinyi u. 18., Nagykanizsa, 8800, Hungary
{bakon.krisztian, holczinger.tibor}@pen.uni-pannon.hu

Abstract. *This paper presents a conceptual reactive scheduling framework for Flexible Job Shops with dynamic order arrivals, due-date targets, and intermediate storage constraints (NIS/UIS). Unlike existing models that treat these aspects separately, our approach integrates them into a unified multi-objective optimization method. We introduce a storage-aware branch-and-bound algorithm with four policy-based flexibility levels and prove a theorem enabling intermediate storage minimization through local slack redistribution. This framework offers a novel way to balance due-date penalties and storage costs, setting the foundation for future real-time applications in adaptive manufacturing environments.*

Keywords. Reactive Scheduling, Flexible Job Shop Problem, Due-Date Optimization, Intermediate Storage Policies, Branch-and-Bound Algorithm, Earliness/Tardiness Minimization, Storage-Gap Adjustment, Policy-Driven Rescheduling

1 Introduction

Modern production systems face challenges from dynamic order arrivals, stringent due dates, and complex resource constraints. This paper presents a comparative framework for reactive (Bakon and Holczinger, 2024) and deterministic scheduling (Bakon and Holczinger, 2025) in an extended Flexible Job Shop Problem (FJSP), integrating real-time order processing, due-date optimization, and intermediate storage policies (NIS/UIS).

Although reactive scheduling and due-date optimization with intermediate storage constraints have each been addressed individually in prior research, they are rarely combined within a single, unified framework—particularly under dynamic order arrivals and rescheduling scenarios. Existing methods typically either overlook the impact of storage behavior on reactive scheduling or focus solely on makespan or tardiness, without incorporating storage-related penalties.

The proposed framework targets a dual objective: minimizing total earliness/tardiness (E/T) penalties and reducing intermediate storage time (IST). Success is defined as achieving a schedule that balances these competing goals under different levels of rescheduling flexibility and realistic storage constraints.

This paper addresses this gap by proposing a conceptual scheduling framework that integrates reactive rescheduling with due-date and storage-aware objectives in an extended FJSP environment. The core contributions are: (1) a policy-aware branch-and-bound (B&B) algorithm supporting four levels of rescheduling flexibility, and (2) the Storage-Gap Adjustment Theorem, which formally enables IST minimization through local slack redistribution. Together, these contributions bridge previously disjoint research threads and lay the foundation for scalable, multi-objective scheduling in adaptive manufacturing systems.

2 Literature Review

Scheduling research spans reactive rescheduling, due-date control, and intermediate storage modeling, yet these dimensions are rarely treated together in a unified framework. However, most contributions treat these dimensions in isolation. This section reviews five thematic strands and clarifies how they inform our unified, policy-driven framework.

Reactive Scheduling Models

Reactive scheduling literature emphasizes balancing responsiveness and stability. Early frameworks distinguish dispatching, predictive–reactive, and proactive-reactive strategies (Bahroun et al., 2024; Vieira et al., 2003). Event-driven policies outperform periodic rescheduling in turbulent environments (Ouelhadj and Petrovic, 2009). More recent contributions incorporate robustness, decision trees (Portoleau et al., 2020), rolling-horizon control laws (Kopanos and Pistikopou-

los, 2014), and adaptive intervals (Wang et al., 2017). Despite this evolution, cognitive constraints and integration with domain-specific policies (e.g., storage constraints) remain under-addressed (Herroelen and Leus, 2005)—a gap this work targets explicitly.

Due-Date Assignment and Flexibility

Due-date strategies range from static job-level rules (e.g., TWK, DPPW) to dynamic updates based on real-time shop load (Cheng and Gupta, 1989; Cheng and Jiang, 1998; T et al., 2017). Extensions incorporate weighted tardiness (Koulamas, 2011), job-shop adaptation (Ojstersek et al., 2020), and project-based economic criteria (Vanhoucke, 2002). While several models address tightness vs. service trade-offs, interactions between due-date pressure and storage-aware rescheduling are rarely modeled. Our framework explicitly incorporates this interaction.

Storage Constraints in Scheduling

Storage modeling has matured in batch and job-shop contexts, formalizing UIS, NIS, and ZW policies (Ha et al., 2000; Kim et al., 1996; Kopanos and Puigjaner, 2009; Romero et al., 2004). S-graph methods and MILP formulations support comparative makespan studies. However, most models are static and ignore rescheduling flexibility. Our policy-aware approach embeds UIS/NIS behavior directly into dynamic rescheduling.

Multi-Objective Scheduling Trends

Recent literature emphasizes Pareto-optimal trade-offs among cost, stability, and service (Holguin Jimenez et al., 2024; Wang et al., 2017). Metaheuristics (e.g., MOSM (Mihály and Kulcsár, 2023), PRIMP (Mansini et al., 2023)) enhance solution diversity, though few frameworks jointly optimize E/T and storage. We address this by integrating both into a reactive branch-and-bound strategy.

Computational Considerations

Scheduling complexity is well established: even simplified job-shop problems are NP-hard with no constant-factor approximations (Mastrolilli and Svensson, 2011). Instance features like dispersion influence heuristic selection (Ruiz-Vanoye et al., 2011). Robust project scheduling models address both solution quality and deviation limits (Herroelen and Leus, 2004, 2005), motivating scalable yet adaptable algorithms. Our model responds with policy-aware pruning and tractable bounding.

In summary, while each domain shows strong development, their integration remains limited. Our contribution lies in combining reactive rescheduling, due-date compliance, and intermediate storage optimization into a single multi-objective scheduling framework for dynamic job-shop environments.

3 Problem Definition

The production environment consists of a finite set of equipment units J, a finite set of products P, and a finite set of tasks I. Each product $p \in P$ is defined by a set of tasks $I_p \subseteq I$, with precedence constraints captured by $I_i^- \subseteq I$, the set of immediate prerequisites for task $i \in I$. Each task $i \in I$ may be performed by a subset of equipment $J_i \subseteq J$, with processing time $pt_{i,j}$ depending on the assigned unit $j \in J_i$.

The production system operates under a dynamic arrival process, where orders $O = \{o_1, o_2, \ldots, o_n\}$ aare revealed sequentially over time. Each order $o_k = (I_k, t_k)$ consists of a product-specific task set $I_k \subseteq I$ and an associated arrival time t_k. All arrival times are assumed distinct, with $t_1 = 0$ without loss of generality. This model does not assume any probabilistic distribution over arrivals; instead, it responds deterministically to each new order as it occurs. Thus, while future arrivals are unknown, the system handles them online in a non-anticipative fashion. Let $I_k^* = \bigcup_{k' \leq k} I_{k'}$ denote the cumulative set of tasks from the first k orders.

Each product $p \in P$ is associated with a due date d_p, either explicitly specified or computed based on the task processing time lower bound:

$$d_p = \left\lfloor f \sum_{i \in I_p} \min_{j \in J_i} pt_{i,j} \right\rfloor .$$

At time t_{k+1}, when a new order o_{k+1} arrives, the current schedule S_k is partitioned into:

- $\underline{S}_k$: the set of assignments that have started before t_{k+1}, and are preserved as fixed assignments in the revised schedule;
- $\overline{S}_k$: the set of assignments that have not yet started and may need to be rescheduled.

Let $\underline{I}_k^*$ and $\overline{I}_k^*$ be the corresponding task sets, such that $I_k^* = \underline{I}_k^* \cup \overline{I}_k^*$.

The objective is to construct a new feasible schedule S_{k+1} satisfying the following:

- All tasks from $I_{k+1} \cup \overline{I}_k^*$ are rescheduled, maintaining precedence and resource constraints;
- Assignments in $\underline{S}_k$ remain unchanged, i.e., $\underline{S}_k \subseteq S_{k+1}$;
- No task begins before its order's arrival time: $t^s \geq t_{k+1}$ for all new assignments;
- The total weighted penalty for E/T across all products up to o_{k+1} is minimized:

$$\min \sum_{p \in P} (w_p^E E_p + w_p^T T_p),$$

where $E_p = \max(0, d_p - C_p)$, $T_p = \max(0, C_p - d_p)$, and C_p is the completion time of the last task $\hat{i}^p$ of product p.

The scheduling environment adheres to one of two intermediate storage policies, each imposing distinct temporal constraints on task execution and inter-operation coordination:

- **NIS**: $\forall i \in I, \quad t^s_{i^+} = t^f_i$ (if equipment available)
- **UIS**: $\forall i \in I, \quad t^s_{i^+} \geq t^f_i$

This problem formulation integrates dynamic order arrivals, task-equipment flexibility, precedence constraints, E/T penalties, and realistic storage behavior into a unified scheduling model. The aim is to develop a robust, scalable, and efficient solution method capable of addressing both the reactivity and optimization challenges in modern production systems.

4 Methodology: Reactive Scheduling with Due-Date and Storage-Aware Optimization

4.1 Reactive Scheduling Framework

Upon receipt of the first order $o_1 = (I_1, t_1 = 0)$, an initial schedule S_1 is constructed. This schedule minimizes due-date deviation (E/T) and respects either the NIS or UIS storage policy. It is derived using the traditional S-graph approach—either equipment-based or task-based branching—extended to compute start and finish times that align as closely as possible with each job's due date d_p.

When a new order $o_{k+1} = (I_{k+1}, t_{k+1})$ arrives, the current schedule S_k is partitioned into:

$$\underline{S}_k = \{(i, j, t^s, t^f) \in S_k \mid t^s < t_{k+1}\}, \quad \overline{S}_k = S_k \backslash \underline{S}_k,$$

with corresponding task sets $\underline{I}^*_k$ (started) and $\overline{I}^*_k$ (not yet started).

Any revised schedule S_{k+1} must ensure that the updated schedule S_{k+1} adheres to the following criteria:

1. Retain all assignments in $\underline{S}_k$.
2. Ensure that no task of I_{k+1} begins before time t_{k+1}.
3. Preserve precedence, storage, and machine-task feasibility.
4. Continue minimizing total $\sum(w^E_p E_p + w^T_p T_p)$, evaluating completion times C_p under either NIS or UIS storage constraints.

4.2 Reactive Policies under Due-Date Minimization

Depending on the desired trade-off between computational efficiency and schedule quality, the rescheduling policy for $\overline{I}^*_k$ can be selected from the following levels of flexibility:

- **Policy 1:** $\overline{I}^*_k$ remains unaltered. New tasks I_{k+1} are appended. While simple and fast, delays incurred by new orders may increase tardiness penalties.
- **Policy 2.1:** Sequencing of $\overline{I}^*_k$ is fixed, and start times are unchanged. New tasks can only be inserted into idle time windows. This strategy safeguards due-date alignment but may limit flexibility if idle time is scarce.
- **Policy 2.2:** Sequence is fixed, but start times of $\overline{I}^*_k$ may shift, facilitating insertions without reassignments. This improves due-date optimization at the cost of modest computation.
- **Policy 3:** Full rescheduling of $\overline{I}^*_k \cup I_{k+1}$ is permitted. This offers the greatest flexibility for due-date equality and storage time reduction but is the most computationally intensive.

4.3 S-Graph Augmentation for Reactive Due-Date Scheduling

The S-graph for rescheduling must encode both the due-date objective and the constraints imposed by reactive policies:

- An artificial root node Z is introduced to connect to each task via arcs weighted by their respective order arrival times, thereby enforcing earliest start time constraints.
- Zero-wait arcs fix the start times of tasks in $\underline{I}^*_k$.
- According to the chosen policy:
  - **Policy 1 & 2.1:** Add zero-wait arcs to all tasks in $\overline{I}^*_k$, freezing their start times.
  - **Policy 2.2:** Only add sequence-preserving schedule-arcs among $\overline{I}^*_k$, allowing temporal flexibility.
  - **Policy 3:** Do not add any arcs among $\overline{I}^*_k$; full rescheduling is allowed.

The graph structure supports integration of task precedence, equipment assignment, and temporal restrictions. B&B search traverses assignments and sequencing steps to build schedule S_{k+1}:

- Each node corresponds to a partial S-graph including current assignments and arcs.
- The algorithm evaluates completion times C_p for all affected products and computes new E_p, T_p, along with storage delays if applicable.

- Lower bounds on the total weighted due-date penalty are computed to prune dominated branches.
- Feasible extensions—such as task insertions or reassignment—are performed in accordance with the constraints imposed by the selected reactive policy.

4.4 Multi-Objective Optimization: Earliness/Tardiness and Intermediate Storage

The reactive scheduling problem involves determining an updated schedule S_{k+1} upon the arrival of a new order o_{k+1} at time t_{k+1}, such that E/T and IST across all jobs are minimized. The multi-objective function is:

$$\min\left(\alpha \cdot IS_{\text{total}} + \beta \cdot (E_{\text{total}} + T_{\text{total}})\right),$$

where $\alpha, \beta \geq 0$ and $\alpha + \beta = 1$. The E/T penalties and IST are computed as:

$$E_{\text{total}} = \sum_{p \in P} \max(0, d_p - t^f_{i^p}),$$
$$T_{\text{total}} = \sum_{p \in P} \max(0, t^f_{i^p} - d_p),$$
$$IS_{\text{total}} = \sum_{i \in I} is_i.$$

4.4.1 Storage-Aware Schedule Adjustments

At each branching step within the S-graph framework, tasks are assigned with respect to both machine availability and storage restrictions. The system checks:

- Recipe sequence constraints for immediate downstream operations
- Machine sequence feasibility without conflicts
- Potential for intermediate storage reduction while maintaining feasibility

If a task i incurs positive IST $is_i > 0$, the scheduler investigates if $is_i = 0$ can yield a valid schedule. If not, a search procedure identifies the minimal feasible is_i, applied recursively to neighboring tasks to balance cumulative IST.

4.5 Algorithmic Framework: Reactive B&B with Storage Optimization

This subsection introduces a storage-aware branch-and-bound (B&B) algorithm tailored to reactive scheduling with due-date and intermediate storage constraints. The algorithm dynamically adjusts its branching strategy based on the chosen rescheduling policy $\pi \in \{1, 2.1, 2.2, 3\}$, balancing computational complexity and scheduling flexibility.

Algorithm 1: Reactive B&B with Due-Date and Storage-Aware Objectives

Input: Current schedule S_k; new order o_{k+1}; policy $\pi \in \{1, 2.1, 2.2, 3\}$
Output: Revised schedule S_{k+1}

Initialize queue Q with root node based on S_k and policy π
Best objective $Z^\star \leftarrow \infty$, $S_{k+1} \leftarrow \emptyset$
while $Q \neq \emptyset$ **do**
  node $u \leftarrow$ pop_best(Q)
  if *LB*$(u) \geq Z^\star$ **then**
    continue
  if u *is complete* **then**
    $Z^\star \leftarrow Z(u)$, $S_{k+1} \leftarrow$ schedule(u)
  else
    Choose unscheduled task i according to branching policy π
    foreach $j \in J_i$ **do**
      $(t^s_i, t^f_i) \leftarrow$ ComputeTentativeInterval(i, j, π, u)
      if (t^s_i, t^f_i) *feasible* **then**
        $is_i \leftarrow$ EstimateStorageTime(i, π)
        if *equipment conflict occurs* **then**
          StorageGapAdjust(i, j)
        Create child node v with updated schedule and costs
        Compute partial objective: $Z(v) \leftarrow \alpha \cdot$ IST$(v) + \beta \cdot$ DueDatePenalty(v)
        Compute lower bound: LB(v)
        if *LB*$(v) < Z^\star$ **then**
          push node v into Q
return S_{k+1}

The following pseudocode (Algorithm 1) provides a structured overview of the algorithm's logic, illustrating the interactions between branching decisions, storage policies, and multi-objective optimization.

The algorithm integrates three core concerns: (i) reactivity to incoming orders by updating the schedule in real time, (ii) due-date adherence via E/T penalties in the objective, and (iii) intermediate storage control through IST estimation and slack redistribution. The function `StorageGapAdjust` ensures local feasibility and cost reduction under UIS policies, while the policy-driven branching logic ensures alignment with practical flexibility constraints.

Complexity and Convergence. The theoretical worst-case complexity of the B&B algorithm grows factorially with the number of reschedulable tasks ($|\overline{I}_k^*|$). Under Policy 1, complexity simplifies to equipment assignments only, yielding $\mathcal{O}((m)^{|\overline{I}_k^*|})$. Policy 3, permitting complete resequencing and reassignment, expands complexity significantly to $\mathcal{O}(|\overline{I}_k^*|!\, m^{|\overline{I}_k^*|})$. However, in practice, empirical convergence is commonly observed within acceptable computational timeframes for realistic scenarios (10–50 tasks, 5–10 machines), aided by tight bounding, heuristic guidance, and effective pruning.

4.5.1 Node Expansion and Temporal Assignment

Each node in the search tree extends the current partial schedule by tentatively assigning a start time t_i^s to an unscheduled task $i \in \overline{I}_k^* \cup I_{k+1}$, respecting precedence ($t_i^s \geq \max(t_{i'}^f)$ for all $i' \in I_i^-$), equipment availability (unit $j \in J_i$ must be idle), order arrival constraints ($t_i^s \geq t_{k+1}$ for new orders), and the immutability of previously fixed assignments in $\underline{S}_k$. The finish time is then set as $t_i^f = t_i^s + pt_{i,j}$. IST is estimated as $is_i = t_{i^+}^s - t_i^f$, with the algorithm attempting to enforce $is_i = 0$ under NIS where possible, and minimizing is_i under UIS via trade-offs between early starts and storage overhead. If conflicts arise on equipment j, the algorithm invokes a storage-gap adjustment: it identifies temporal neighbors i^- and i^+, redistributes slack ($\delta > 0$) from their respective storage times, and verifies that precedence remains valid ($t_{i^+}^s \geq \max(t_{i'}^f)$). If adjustment fails, t_i^s is incremented and reassessed. After placement, the algorithm evaluates the partial objective function,

$$Z = \alpha \sum_{i \in I_{\text{sched}}} is_i + \beta \sum_{p \in P_{\text{sched}}} (E_p + T_p),$$

where I_{sched} and P_{sched} are the sets of scheduled tasks and completed products, respectively, and $E_p = \max(0, d_p - t_{i_p}^f)$, $T_p = \max(0, t_{i_p}^f - d_p)$ are early/tardy penalties. Finally, nodes are pruned if their lower bound $Z_{\text{lower bound}}$, estimated using heuristics and policy-specific constraints, exceeds the current best objective value.

4.5.2 Storage-Gap Adjustment Proposition

Proposition 1. *For any feasible UIS schedule, local slack redistribution can strictly reduce the total intermediate storage time IS_{total} without violating precedence or resource constraints.*

Proof. Let $\Gamma = \{i_1, \ldots, i_q\}$ be the tasks on a unit j ordered by start time. Define $\Delta_k = t_{i_{k+1}}^s - t_{i_k}^f \geq 0$. If $\sum_k \Delta_k = 0$ the schedule is no-wait and thus minimal. Otherwise select $k^* = \arg\max_k \Delta_k$ and shift tasks $i_{k^*+1}, \ldots, i_q$ left by $\varepsilon = \min\{\Delta_{k^*}, \min_{r>k^*}(\Delta_r + pt_{i_r})\}$. All predecessor finish times remain $\leq t_{i_{k^*+1}}^s - \varepsilon$, so precedence holds. Each iteration reduces IS_{total} by ε; after at most $q-1$ iterations $\sum_k \Delta_k = 0$. □

This proposition guarantees a local improvement in intermediate storage under the UIS assumption. However, it does not constitute a proof of global optimality or convergence, which remains an open challenge for future work.

4.5.3 Lower Bounding Techniques

Accelerate search via:

- **Zero-Lag Bound**: Optimistic assumption:

  $$C_p = d_p \quad \text{and} \quad is_i = 0 \quad \forall \text{ unscheduled } i, p$$

  Provides aggressive pruning for early-stage nodes
- **Critical-Path Bound**: Computes longest delay path through precedence arcs:

  $$\text{CP}_{\text{delay}} = \max_{\text{paths}} \left(\sum_{(i',i)} (t_i^s - t_{i'}^f) \right)$$

  Incorporates temporal and storage latency.
- **IST Heuristic**: For unscheduled task i, estimate minimal feasible is_i considering:
  - Temporal proximity to immediate predecessors
  - Projected equipment availability
  - Policy constraints (NIS/UIS)

4.5.4 Multi-Objective Trade-Off and Dominance Consideration.

Although Pareto frontiers are not empirically constructed here, the scalarized cost function

$$Z = \alpha \sum_{i \in I_{\text{sched}}} is_i + \beta \sum_{p \in P_{\text{sched}}} (E_p + T_p)$$

implies an underlying trade-off surface between IST and due-date penalties. Varying α and β across runs could approximate a frontier. Since both objectives may conflict, future implementations could apply dominance checks to improve pruning. The non-convexity of this space—due to discrete task sequencing—also suggests that metaheuristic or adaptive scalarization techniques may better approximate trade-offs in practice.

4.6 Policy-Aware Search Adaptation and Computational Complexity Analysis

The proposed B&B algorithm dynamically adjusts its search strategy based on the selected reactive rescheduling policy, directly influencing branching complexity and computational requirements. Let $n = |\overline{I}_k^* \cup I_{k+1}|$ be the number of unscheduled tasks and $m = |J|$ the number of machines. Each policy imposes distinct constraints:

- **Policy 1 (Append-only):** Tasks in $\overline{I}_k^*$ remain fixed in sequence and timing. New tasks I_{k+1} are simply appended. This results in low branching complexity $O(nm)$, due to the linear number of new task-machine assignment decisions. However, limited flexibility often results in suboptimal solutions concerning due-date and intermediate storage objectives.
- **Policy 2.1 (Idle-Gap Insertion):** The sequence and start times of tasks in $\overline{I}_k^*$ remain fixed, but new tasks can be inserted into available idle time windows. This moderately increases flexibility and computational complexity to $O(n^2m)$ due to the polynomial number of task-slot-machine checks required. The policy balances due-date adherence but may restrict task placement significantly when idle time is scarce.
- **Policy 2.2 (Fixed Sequence, Shifted Starts):** Tasks in $\overline{I}_k^*$ retain their sequence but can shift their start times within constraints. This additional temporal flexibility significantly improves the potential for optimal due-date alignment, increasing branching complexity to approximately $O(n^3m)$, influenced by temporal discretization and resource availability.
- **Policy 3 (Full Rescheduling):** Offers maximum flexibility, allowing full resequencing and reassignment of all unscheduled tasks $\overline{I}_k^* \cup I_{k+1}$. This combinatorial freedom leads to factorial complexity $O(n! \cdot m^n)$, making it computationally intensive but optimal in solution quality.

The selected policy fundamentally shapes algorithm performance and suitability for real-time implementation:

- **Branching Complexity:** Ranges from linear (Policy 1) to factorial (Policy 3).
- **Constraint Handling:** Policies 1 and 2 enforce hard constraints, significantly reducing flexibility.
- **Solution Quality:** Flexibility directly correlates with the potential to optimize E/T penalties and IST.
- **Real-Time Viability:** Lower-complexity policies (1, 2.1) offer rapid solutions suitable for real-time scenarios but sacrifice optimality, whereas higher-complexity policies (2.2, 3) achieve better optimization at increased computational cost.

Although the underlying combinatorial complexity remains fundamentally invariant to the chosen objective functions, incorporating multi-objective criteria—such as E/T and IST—affects the algorithm's practical efficiency. Specifically, multi-objective considerations influence the effectiveness of lower-bounding techniques and branch pruning, resulting in increased computational overhead under highly flexible policies (e.g., Policy 3) despite identical theoretical complexity.

5 Discussion on Trade-offs

The proposed reactive scheduling framework inherently balances multiple competing factors, such as flexibility, computational complexity, due-date adherence (E/T penalties), and storage costs. Table 1 succinctly illustrates these trade-offs across the defined rescheduling policies.

Table 1. Qualitative Trade-off Comparison Across Rescheduling Policies

Policy	Flexibility	Comp. Time	E/T Penalty	Storage Penalty
1	Low	Low	High	High
2.1	Medium	Medium	Moderate	Moderate
2.2	High	Med–High	Low	Moderate
3	Very High	High	Lowest	Lowest

Policy 1 is suitable for environments prioritizing computational efficiency and scheduling stability, yet its limited flexibility typically results in higher penalties for both due-date adherence and intermediate storage. Policies 2.1 and 2.2 incrementally enhance flexibility, progressively improving the schedule's alignment with due-date targets while moderately controlling storage costs at the expense of increased computational complexity. Policy 3 offers maximal flexibility, achieving the lowest penalties for both due-date violations and intermediate storage. However, this optimality comes at the highest computational cost, making it best suited for scenarios where solution quality outweighs computational constraints.

Future modelling extensions should explore more nuanced scenarios, including hybrid or adaptive storage policies, dynamic task prioritization, and robustness analysis under stochastic disruptions, to deepen insights into these complex trade-offs.

6 Conclusion and Future Work

This paper establishes a conceptual foundation for integrating reactive scheduling with due-date management and intermediate storage constraints in flexible job shop environments. The proposed framework introduces a multi-objective branch-and-bound algorithm that balances E/T penalties and IST, structured around

four reactive rescheduling policies that progressively increase scheduling flexibility.

A key theoretical contribution is the Storage-Gap Adjustment Proposition, which provides a constructive approach for local IST minimization under UIS constraints via targeted slack redistribution. While this result offers promising structure-preserving improvement, it remains conceptual and lacks empirical implementation at this stage.

Importantly, the framework evaluates candidate schedules using a weighted cost function combining storage and due-date objectives. While scalarized in this study, the underlying trade-offs are inherently multi-objective. Varying the weighting coefficients would yield a Pareto frontier of non-dominated schedules, with each representing a distinct balance of E/T performance and storage cost. Future work will explore these frontiers explicitly, assessing convexity and dominance relationships to guide solution selection. As the objective space may be non-convex due to combinatorial constraints, such analysis can inform adaptive weight adjustment or hybrid optimization strategies.

Future research should also focus on computational benchmarking using simulated or industrial instances to evaluate policy-specific scalability and solution quality. Additionally, extending the model to incorporate uncertainty—such as stochastic order arrivals, equipment failures, and variable processing times—would improve robustness. Finally, integrating real-time decision support elements such as digital twins, reinforcement learning for policy selection, and cost-aware storage models would further enhance the framework's practical applicability in dynamic manufacturing environments.

Nomenclature

α, β	Multi-objective weights ($\alpha + \beta = 1$)
$\overline{I}_k^*$	Tasks in $\overline{S}_k$
$\overline{S}_k$	Unstarted assignments (reschedulable)
$\underline{I}_k^*$	Tasks in $\underline{S}_k$
$\underline{S}_k$	Started assignments (fixed) before t_{k+1}
C_p	Completion time of product p's last task
d_p	Due date for product p
$E_p = \max(0, d_p - C_p)$	Earliness of product p
I	Set of tasks
$I_i^- \subseteq I$	Immediate prerequisites for task i
$I_p \subseteq I$	Tasks for product p
is_i	Intermediate storage time for task i
J	Set of equipment units
$J_i \subseteq J$	Equipment capable of performing task i
O	Set of orders $\{o_1, o_2, \ldots, o_n\}$
P	Set of products
$pt_{i,j}$	Processing time of task i on unit j
S_k	Schedule at order k's arrival
t_i^s, t_i^f	Start/finish time of task i
t_k	Arrival time of order o_k
$T_p = \max(0, C_p - d_p)$	Tardiness of product p
w_p^E, w_p^T	Earliness/tardiness weights for product p

References

Bahroun, Z., Shamayleh, A., As'ad, R., & Zakaria, R. (2024). Integrated proactive-reactive tool for dynamic scheduling of parallel machine operations. *International Journal of Engineering Business Management*, *16*, 18479790241301164. https://doi.org/10.1177/18479790241301164

Bakon, K. A., & Holczinger, T. (2024). S-graph-based reactive scheduling with unexpected arrivals of new orders. *Machines*, *12*(7). https://doi.org/10.3390/machines12070446

Bakon, K. A., & Holczinger, T. (2025). Addressing due date and storage restrictions in the s-graph scheduling framework. *Machines*, *13*(2). https://doi.org/10.3390/machines13020131

Cheng, T., & Gupta, M. (1989). Survey of scheduling research involving due date determination decisions. *European Journal of Operational Research*, *38*(2), 156–166. https://doi.org/https://doi.org/10.1016/0377-2217(89)90100-8

Cheng, T., & Jiang, J. (1998). Job shop scheduling for missed due-date performance. *Computers & Industrial Engineering*, *34*(2), 297–307. https://doi.org/https://doi.org/10.1016/S0360-8352(97)00317-3

Ha, J.-K., Chang, H.-K., Lee, E. S., Lee, I.-B., Lee, B. S., & Yi, G. (2000). Intermediate storage tank operation strategies in the production scheduling of multi-product batch processes. *Computers & Chemical Engineering*, *24*(2), 1633–1640. https://doi.org/https://doi.org/10.1016/S0098-1354(00)00438-5

Herroelen, W., & Leus, R. (2004). Robust and reactive project scheduling: A review and classification of procedures. *International Journal of Production Research*, *42*(8), 1599–1620.

Herroelen, W., & Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials [Project Management and Scheduling]. *European Journal of Operational Research*, *165*(2), 289–306. https://doi.org/https://doi.org/10.1016/j.ejor.2004.04.002

Holguin Jimenez, S., Trabelsi, W., & Sauvey, C. (2024). Multi-objective production rescheduling: A systematic literature review. *Mathematics*, *12*(20), 3176.

Kim, M., Jung, J. H., & Lee, I.-B. (1996). Optimal scheduling of multiproduct batch processes for various intermediate storage policies. *Industrial & Engineering Chemistry Research*, *35*, 4058–4066. https : / / api . semanticscholar . org / CorpusID : 96817294

Kopanos, G. M., & Pistikopoulos, E. N. (2014). Reactive scheduling by a multiparametric programming rolling horizon framework: A case of a network of combined heat and power units. *Industrial & Engineering Chemistry Research*, *53*(11), 4366–4386. https://doi.org/10.1021/ie402393s

Kopanos, G. M., & Puigjaner, L. (2009). A milp scheduling model for multi-stage batch plants. In *Computer aided chemical engineering* (pp. 369–374, Vol. 26). Elsevier.

Koulamas, C. (2011). A unified solution approach for the due date assignment problem with tardy jobs. *International Journal of Production Economics*, *132*(2), 292–295.

Mansini, R., Zanella, M., & Zanotti, R. (2023). Optimizing a complex multi-objective personnel scheduling problem jointly complying with requests from customers and staff. *Omega*, *114*, 102722.

Mastrolilli, M., & Svensson, O. (2011). Hardness of approximating flow and job shop scheduling problems. *Journal of the ACM (JACM)*, *58*(5), 1–32.

Mihály, K., & Kulcsár, G. (2023). A new many-objective hybrid method to solve scheduling problems. *International Journal of Industrial Engineering and Management*, *14*(4), 326–335.

Ojstersek, R., Tang, M., & Buchmeister, B. (2020). Due date optimization in multi-objective scheduling of flexible job shop production. *Advances in Production Engineering & Management*, *15*(4), 481–492.

Ouelhadj, D., & Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems. *Journal of scheduling*, *12*, 417–431.

Portoleau, T., Artigues, C., & Guillaume, R. (2020). Robust predictive-reactive scheduling: An information-based decision tree model. *Information Processing and Management of Uncertainty in Knowledge-Based Systems: 18th International Conference, IPMU 2020, Lisbon, Portugal, June 15–19, 2020, Proceedings, Part III*, *1239*, 479–492. https://doi.org/10.1007/978-3-030-50153-2_36

Romero, J., Puigjaner, L., Holczinger, T., & Friedler, F. (2004). Scheduling intermediate storage multipurpose batch plants using the s-graph. *AIChE Journal*, *50*(2), 403–417.

Ruiz-Vanoye, J. A., Díaz-Parra, O., & Zavala-Díaz, J. C. (2011). Complexity indicators applied to the job shop scheduling problem to discriminate the best algorithm. *International Journal of Combinatorial Optimization Problems and Informatics*, *2*(3), 25–31.

T, V. K., A, J. O., & Setupathi, R. (2017). Multi-objective comparison of due-date assignment methods in a dynamic job shop with sequence dependent setup time. *International Journal of Engineering Research & Technology (IJERT)*, *6*(04).

Vanhoucke, M. (2002, December). *Optimal Due Date Assignment In Project Scheduling* (Working Papers of Faculty of Economics and Business Administration, Ghent University, Belgium No. 02/159). Ghent University, Faculty of Economics and Business Administration.

Vieira, G. E., Herrmann, J. W., & Lin, E. (2003). Rescheduling manufacturing systems: A framework of strategies, policies, and methods. *Journal of scheduling*, *6*, 39–62.

Wang, L., Luo, C., & Cai, J. (2017). A variable interval rescheduling strategy for dynamic flexible job shop scheduling problem by improved genetic algorithm. *Journal of Advanced Transportation*, *2017*(1), 1527858.