Bottleneck-Oriented Overlap Scheduling for Multi-Level Assembly Systems

Chen-Hong
Graduate Institute of Telecommunications and Communication
National Chung Cheng University, Taiwan
leon28473787@gmail.com

***Abstract*—This study investigates the bottleneck-shifting phenomenon commonly observed in multi-level Bill of Materials (BOM) environments and proposes an event-driven scheduling framework enhanced by a δ-overlap activation mechanism. Traditional static list-based scheduling often struggles with delayed reactions and idle gaps when faced with deep dependency chains and asynchronous material arrivals. To address these challenges, we develop BOS-Greedy, a structured and analytically tractable scheduling method that prioritizes tasks using a dynamic remaining critical-path metric and updates scheduling decisions upon task completion, material arrival, or partial-kit feasibility.**

Experimental results show that, under static multi-level BOM settings, BOS-Greedy with $\delta = 0$ achieves scheduling quality comparable to HEFT, validating its theoretical consistency with classical deterministic scheduling logic. When overlap is allowed ($\delta > 0$), the makespan exhibits a stable and monotonic reduction, with improvements of approximately 0.7%–1.1% without increased variance, indicating that δ-overlap effectively shortens the overlapped critical-chain length L_δ. Under bottleneck deterioration scenarios, BOS-Greedy demonstrates stability equivalent to HEFT and significantly outperforms List Scheduling, confirming that event-driven execution enables rapid adaptation to dependency changes.

We further derive a theoretical upper bound for BOS-Greedy and show that its makespan satisfies

$$C_{\max} \leq \frac{P}{m} + \left(1 - \frac{1}{m}\right) L_\delta,$$

where L_δ denotes the effective critical-chain length under δ-overlap. This result generalizes Graham's classical bound for list scheduling to settings involving partial-kit activation and dynamic bottleneck evolution, while preserving the $(2 - 1/m)$ worst-case approximation ratio.

Overall, BOS-Greedy provides a lightweight, interpretable, and theoretically grounded scheduling framework capable of handling multi-level BOM structures, δ-overlap activation, and bottleneck shifting. The method is suitable for real-time scheduling in AMRP, APS, and Industry 4.0 environments. Future work includes adaptive δ-learning, multi-event bottleneck disturbances, integration of setup times and batching, and scalability analysis under large machine counts and deep BOM structures.

***Index Terms*—Multi-level BOM scheduling, dynamic bottleneck, partial-kit activation, δ-overlap, event-driven scheduling.**

I. Introduction

Modern manufacturing systems rely on multi-level Bills of Materials (BOMs) that exhibit explicit hierarchical struc- tures, which can be naturally modeled as directed acyclic graphs (DAGs) with precedence constraints [1]. Nodes rep- resent procurement or production operations, while edges encode material-availability restrictions and prerequisite re- lations. Such hierarchy-driven dependency models form the analytical foundation of multi-level manufacturing scheduling problems [2], [3]. Classical Material Requirements Planning (MRP) and Advanced MRP (AMRP) systems are widely used to coordinate multi-stage production: MRP determines release times based on demand, inventory, and supply lead times, whereas AMRP incorporates machine capacity, cross-facility coordination, and supply-chain variability [4], [5].

However, the commonly adopted practice of partial assembly introduces substantially greater scheduling complexity than assumed in traditional MRP frameworks [6]. When certain components arrive early, upstream operations may begin immediately, while critical components join only in later stages. Thus, operation initiation is no longer equivalent to requiring all predecessors to be completed. Actual start times are influenced by delay tolerance, discrepancies in material arrivals, and the depth of the BOM hierarchy. In this context, the effective critical path is often dictated by the slowest-arriving bottleneck component, and the bottleneck may shift over time due to supply delays, processing congestion, and capacity competition [7]. Consequently, the critical path is not a fixed longest-path structure [8] but instead evolves dynamically with the allowable overlap δ.

Scheduling with precedence constraints on parallel machines is NP-hard [9]. List Scheduling provides a fundamental approximation guarantee of $(2 - 1/m)$ for such problems [10], forming a cornerstone of classical scheduling theory. Yet prior work shows that local greedy decisions can lead to timing anomalies in multiprocessor environments, where adding resources paradoxically increases completion time [10]. Furthermore, the literature on Resource-Constrained Project Scheduling (RCPSP) indicates that, under resource contention, the effective completion time is governed by the resource-feasible critical chain, rather than the static longest path used by heuristics such as LPT [11]–[13]. Thus, the static longest-path perspective is insufficient in practical systems where bottlenecks are shaped by resource competition, material delays, and workload fluctuations [14], [15].

Approximation frameworks for shop scheduling—such as the ideal-schedule $\rightarrow$ random-delay $\rightarrow$ flattening paradigm—offer polylogarithmic guarantees [16], [17], but they fail to capture cross-level dependencies in multi-level BOMs or asynchronous material arrivals [18]. Similarly,

AND/OR precedence models allow partial predecessor completion but do not explicitly address lead times, hierarchical depth, or delay tolerance, making them unsuitable for multi-level manufacturing scenarios involving partial assembly [19]. Overall, existing literature lacks a unified scheduling framework that simultaneously captures: (i) deep BOM dependencies, (ii) dynamic bottleneck evolution, (iii) overlap-based initiation, and (iv) cross-operation interactions. Moreover, no algorithmic solution currently provides deterministic approximation guarantees under these combined characteristics [20], [21].

In multi-level manufacturing processes, overall completion time is dictated by the dynamic behavior of the bottleneck chain rather than a static longest path. This behavior results from heterogeneous material arrivals, mid-process insertion of critical components, early operation initiation allowed by δ-tolerance, and shifts in bottlenecks caused by fluctuating machine loads [22]. Thus, future production scheduling and control must not rely solely on static rules; they must account for increasingly dynamic and uncertain manufacturing environments.

Although machine learning has been widely considered a promising enhancement to heuristic scheduling, recent surveys emphasize that such techniques cannot yet serve as full-scale replacements for complex decision-making algo- rithms [23]. Their performance depends heavily on specific data distributions or workload patterns and may degrade sharply under environmental shifts. Deep models also require extensive training data and costly offline exploration. Without approxi- mation guarantees or feasibility assurances, learning-based scheduling can yield unpredictable or suboptimal outcomes in highly time-sensitive manufacturing settings—risking increased idle time, longer production cycles, or failure to meet delivery deadlines.

Furthermore, most learning-based scheduling or routing ap- proaches lack interpretability, feasibility guarantees, and the theoretical performance bounds characteristic of classical al- gorithms. Therefore, learning-based approaches are better positioned as predictive auxiliary mechanisms rather than complete substitutes for structured scheduling theory.

Currently, no scheduling method simultaneously:

- captures dynamic bottleneck behavior,
- preserves the provable structure of list scheduling,
- supports δ-overlap initiation, and
- provides deterministic approximation guarantees.

Given these challenges, practical manufacturing systems re- quire a deterministic approach that integrates: (1) structured decision logic, (2) verifiable approximation performance, and (3) explicit handling of overlap-based initiation and bottleneck- chain dynamics.

To this end, we propose the **Bottleneck-Oriented Overlap Scheduling (BOS)** method, built upon the following principles:

- a list-scheduling–based theoretical backbone,
- an overlap-aware initiation mechanism,
- a bottleneck-chain overlap metric for prioritization,
- deterministic, analytically provable approximation guarantees.

II. Related Work

Multi-level manufacturing scheduling involves hierarchical precedence constraints, multi-machine resource competition, cross-level material arrivals, and partial-kit activation. These characteristics cannot be fully captured by a single classical scheduling model. This section reviews four major bodies of literature to identify the theoretical limitations faced when modeling multi-level BOM structures and δ-overlap activation.

A. Problem Setting: Multi-Level BOM, Hierarchical Precedence, and Dynamic Bottlenecks

Modern manufacturing systems commonly employ multi-level Bills of Materials (BOMs) to represent product structures. The parent–child relationships in a BOM naturally form a directed acyclic graph (DAG) [24], [25]. A BOM distinguishes between independent demand for finished goods and dependent demand for subcomponents, and its deep hierarchical structure determines the permissible order of operations across production stages.

However, multi-level BOM scheduling differs fundamentally from classical precedence-constrained parallel-machine models such as $P \parallel prec \parallel C_{max}$. Although List Scheduling provides a $(2 - 1/m)$ approximation bound [10], the model assumes that all tasks are known at time zero and depend solely on logical precedence relations. It does not incorporate material availability, supply lead times, or asynchronous cross-level component arrivals, and therefore represents a flat and static precedence graph.

In contrast, dependencies in multi-level BOMs arise from *material causality*. Higher-level assemblies cannot begin until all subcomponents have been procured, processed, and physically delivered. Thus, start times are shaped by real-world supply delays, queueing effects, processing load variations, and heterogeneous arrival patterns. This leads to a time-sensitive DAG whose behavior is driven by dynamic material availability.

Moreover, bottlenecks in multi-stage production shift over time. In classical scheduling, makespan is often determined by a fixed longest path [1]. However, in multi-level BOM systems, limiting factors may change due to material delays, machine congestion, or temporary queue buildups [26], [27]. Such *bottleneck shifting* implies that the critical chain is dynamic rather than static.

Overall, multi-level BOM scheduling is difficult because:

- operation start times depend on actual material arrivals rather than predetermined release constraints;
- delays in any component may propagate upward across layers;
- bottleneck locations evolve over time due to material flow variability and resource contention.

These characteristics do not exist in standard $P \parallel prec \parallel C_{max}$ models, indicating that multi-level BOM scheduling is not a direct subset of classical precedence-constrained scheduling.

B. Partial-Kit Activation, δ-Overlap, and Asynchronous Material Arrivals

A second key distinction from classical scheduling models is that many manufacturing systems allow operations to begin before all predecessors are completed. Industries such as semiconductor packaging, automotive electronics, and assembly manufacturing often adopt *partial-kit* policies [28]: once a subset of required components arrives, upstream operations start immediately to reduce wait time caused by late-arriving parts. This deviates from the traditional AND-precedence assumption that all predecessors must be completed.

This work formalizes such behavior using a tolerance parameter δ, representing the allowable degree of overlap between missing dependencies and operation initiation. Under δ-overlap activation, an operation may begin as long as the missing components do not block the executable portion. This is neither classical AND-precedence nor OR-precedence [19], but an intermediate dependency that allows limited overlap. The value of δ directly influences the feasible scheduling space.

Material arrivals in multi-level BOM systems are typically asynchronous. Heterogeneous arrival times cause some stages to begin early while others must wait, producing a hybrid execution pattern of "partially overlapping" and "partially blocked" operations.

Existing scheduling frameworks—including $P \parallel prec \parallel C_{max}$, RCPSP [11], and AND/OR models [11], [19]—assume strict activation rules that are not violated. None of them capture the behavior of "partial dependency satisfaction" allowed under δ-overlap. Therefore, current approximation algorithms cannot be directly applied to multi-level scheduling with partial-kit activation.

C. Dynamic Bottlenecks and Critical-Chain Deviation

In multi-stage manufacturing, makespan is not determined by a fixed longest path but by whichever production stage is currently constrained. Bottlenecks may shift due to temporary overloads, queue buildup, machine delays, or late materials [7]. These shifts occur rapidly and unpredictably, meaning the critical chain is not topologically predetermined but dynamically formed by system conditions.

As a result, any scheduling algorithm unable to detect and respond to bottleneck movement risks producing suboptimal or incorrect decisions. Dynamic bottleneck shifting is a structural challenge that stems from multi-stage interactions and is independent of δ-overlap or arrival variability. Static precedence graphs cannot represent this behavior.

D. Classical Deterministic Scheduling Algorithms

1) Method Categories: Deterministic approximation algorithms for precedence scheduling can be grouped into five major categories:

a) 1) Greedy / List Scheduling: Simple, practical, and provides the $(2 - 1/m)$ bound [10]. However, it assumes static precedence and does not support asynchronous arrivals or changing ready sets.

b) 2) HEFT (Heterogeneous Earliest Finish Time): A widely used DAG scheduler for heterogeneous processors [29]. Limitations include:

- all task information known at time zero,
- strict AND-precedence,
- static DAG structure,
- no support for δ-overlap or dynamic bottlenecks.

c) 3) LP Rounding: Offers strong worst-case guarantees such as 2-approx or PTAS [16], [30], but assumes all tasks can be modeled in a fixed precedence graph at time zero—invalid for BOM systems.

d) 4) Primal–Dual / Dual-Fitting: Effective for certain constrained settings but depends on static feasible domains [31].

e) 5) Local Search: Heuristically strong but lacks theoretical guarantees and assumes fixed topology [32].

2) Representative Approximation Ratios:

- Graham List Scheduling: $(2 - 1/m)$
- LP-based: 2-approx or PTAS
- Shmoys–Stein–Wein: polylogarithmic approximations
- Potts & Van Wassenhove: 2–3 approximation for batching and related settings

3) Limitations of Classical Methods: Classical approximation frameworks cannot handle:

- δ-overlap activation,
- asynchronous material arrival,
- dynamic bottleneck movement,
- cross-level delay propagation,
- multi-level BOM causal structures.

Thus, although theoretically mature, they are structurally incompatible with multi-level manufacturing requirements.

E. Machine-Learning-Based Scheduling

Recent ML-based scheduling approaches include reinforcement learning, supervised dispatching models, and learning-to-schedule frameworks [33]–[35]. They excel at pattern recognition and predictive tasks but suffer from:

- no performance guarantees (no approximation ratio),
- catastrophic degradation under workload shifts,
- lack of feasibility enforcement,
- inability to model material causality,
- inability to track bottleneck shifting.

Thus, ML is suitable as an auxiliary predictor, not a core scheduling mechanism for multi-stage BOM systems.

F. Gap Analysis

Method	BOM	δ	Bottleneck	Approx	Feasible
Greedy/LS	×	×	×	✓	✓
LP Rounding	×	×	×	✓	✓
Local Search	×	×	×	✓	✓
ML Scheduling	×	×	×	×	×
BOS (ours)	✓	✓	✓	✓	✓

G. Positioning of BOS: Contributions Over Prior Work

The Bottleneck-Oriented Overlap Scheduling (BOS) method addresses the limitations identified above by providing:

1) a deterministic, interpretable scheduling rule built on analyzable list-scheduling structure;
2) a bottleneck-chain metric D_i extending longest-path reasoning to dynamic environments;
3) native support for multi-level BOM dependencies and material causal structure;
4) explicit modeling of δ-overlap activation;
5) event-driven recomputation capable of tracking dynamic bottleneck shifting;
6) a balance between theoretical analyzability and practical industrial feasibility.

BOS is therefore one of the few existing frameworks capable of jointly handling multi-level BOM structures, δ-overlap activation, and dynamic bottleneck behavior within a deterministic, approximation-aware scheduling paradigm.

H. Approximation Property of BOS-Greedy

To connect the proposed event-driven scheduling model with classical deterministic scheduling theory, we provide a generalized performance bound extending the well-known $2 - \frac{1}{m}$ result of Graham for list scheduling.

Let L_δ denote the δ-overlapped critical-chain length, defined as the longest feasible dependency chain under the δ-tolerant activation model.

[Generalized Graham Bound for BOS-Greedy] For identical parallel machines, the makespan produced by BOS-Greedy satisfies:

$$C_{\max} \leq \frac{1}{m} \sum_{i \in T} p_i \; + \; \left(1 - \frac{1}{m}\right) L_\delta.$$

Proof Sketch. The result follows from two observations:

1) The dynamic bottleneck index $D_i(t)$ ensures that at every event time, BOS-Greedy selects a task on (or close to) a δ-feasible critical chain. Thus the longest overlapped chain L_δ lower-bounds the best achievable schedule length.
2) Machine idleness occurs only when no task satisfies either release-time or δ-activation constraints. Thus, the classical envelope argument used in Graham's analysis applies, yielding the load term $\frac{1}{m} \sum_i p_i$.

Combining the chain lower bound and load balance upper bound yields the stated inequality.

III. Problem Formulation and Theoretical Foundations

This section formalizes the multi-level BOM scheduling problem, the δ-overlap activation rule, the event-driven execution model, and the dynamic bottleneck index used by BOS-Greedy.

A. System Model

Let $T = \{1, \ldots, n\}$ be the set of tasks and $M = \{1, \ldots, m\}$ the set of identical parallel machines. Each task i has a processing time $p_i > 0$, a release time $r_i \geq 0$, and may begin only when both material availability and precedence constraints are satisfied.

Algorithm 1 BOS-Greedy: Summary of Algorithmic Steps

1. Compute D_i for each task (remaining critical-path length; Dynamic Priority).
2. Construct the executable task set $\mathcal{R}$ and organize it using a max-heap.
3. When tasks start or complete:
   - If a new task becomes executable, push it into the heap.
   - If a task completes or is preempted, remove it from the heap.
   - Perform machine assignment and possible preemption.
4. Repeat until all tasks have finished.

Algorithm 2 BOS-Greedy: Event-Driven Scheduling Steps

1. **Initialization:** Construct the initial executable task set $\mathcal{R}$, containing tasks that satisfy release-time and δ-overlap activation conditions.
2. **Priority Queue Construction:** Build a max-heap H over $\mathcal{R}$, using the dynamic priority D_i as the key.
3. **Event-Driven Scheduling:** While unfinished tasks remain:
   3.1 **Wait for the next event** e**.** Possible events include task completion, activation, machine release/recovery, or any event affecting feasibility.
   3.2 **Update task states:** Recompute D_i for affected tasks and update $\mathcal{R}$.
   3.3 **Update the heap:** Push newly activated tasks; remove completed or preempted tasks; reorder if priorities change.
   3.4 **Machine assignment:** Assign idle machines to tasks with the highest D_i; perform legal preemption if necessary.
4. **Termination:** Stop when all tasks have completed.

1) BOM and Precedence Structure: A multi-level Bill of Materials induces a directed acyclic graph (DAG) $G = (T, E)$, where $(j, i) \in E$ indicates that i depends on the completion of j. Let $\text{pred}(i)$ be the set of immediate predecessors.

2) Feasible Scheduling: A schedule assigns each task i a start time s_i and finish time $f_i = s_i + p_i$, together with a machine assignment $\mu(i)$. Feasibility requires the following constraints:

$$s_i \geq r_i, \tag{1}$$

$$s_i \geq \max_{j \in \text{pred}(i)} f_j, \tag{2}$$

The classical precedence condition in (2) enforces that all predecessors of task i must complete before i may begin. In addition, no machine may process two tasks simultaneously:

$$\mu(i) = \mu(j) \;\Rightarrow\; [s_i, f_i] \cap [s_j, f_j] = \emptyset. \tag{3}$$

The objective is to minimize the makespan $C_{\max} = \max_i f_i$.

B. δ-Overlap Activation

Real manufacturing often initiates tasks before all predecessors finish. Let the dependency completion ratio at time t be:

$$\Phi_i(t) = \frac{|\{j \in \text{pred}(i) \;:\; f_j \leq t\}|}{|\text{pred}(i)|}.$$

[δ-Overlap Activation] Task i may start at time t when

$$\Phi_i(t) \geq 1 - \delta, \qquad \delta \in [0, 1].$$

Thus,

$$s_i \geq r_i, \qquad \Phi_i(s_i) \geq 1 - \delta.$$

[Monotonicity] For any $0 \leq \delta_1 < \delta_2 \leq 1$, the feasible overlapped-chain length satisfies

$$L_{\delta_2} \;\leq\; L_{\delta_1}.$$

A larger δ relaxes the activation condition $\Phi_i(t) \geq 1 - \delta$, enlarging the feasible region. Thus every δ_1-feasible chain is also δ_2-feasible. Taking the longest chain on both sides preserves the inequality.

[Equivalence at $\delta = 0$] When $\delta = 0$, the overlapped-chain length reduces to the classical critical-path length:

$$L_0 = L_{\text{CP}}.$$

At $\delta = 0$, activation requires all predecessors to finish. Thus P_0 is exactly the AND-precedence longest path.

C. Event-Driven Scheduling Model

Let

$$\mathcal{E} = \{\text{completion, activation, machine release, material arrival}\}.$$

At an event e, the system state is:

$$\mathcal{S}(e) = \big(H(e),\, \mathcal{R}(e),\, \text{running}(e)\big).$$

State transitions apply:

$$\mathcal{S}(e^+) = \mathcal{T}\big(\mathcal{S}(e^-), e\big),$$

where $\mathcal{T}$ updates dependency ratios, checks δ-activation, updates the ready set, adjusts heap keys D_i, and applies preemption.

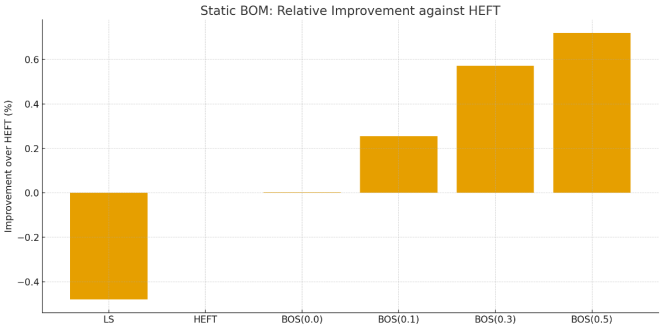

Fig. 1. Static BOM: Relative improvement against HEFT.

D. Dynamic Bottleneck Index

For any path $P = (i, j_1, j_2, \dots)$ define

$$L(P) = \sum_{k \in P} p_k.$$

[Dynamic Bottleneck Index]

$$D_i(t) = \max_{P \in \mathcal{P}(i)} L(P).$$

At any event time t, BOS-Greedy selects a task lying on a prefix of a δ-feasible critical chain P_δ.

By definition,

$$D_i(t) = \max_{P \in \mathcal{P}(i)} L(P),$$

so a task attains maximal priority iff it lies on a path of maximal remaining load. Since P_δ has maximal feasible load under δ-activation, its prefix contains all tasks achieving the maximum $D_i(t)$. BOS-Greedy always selects such a task.

E. Time Complexity

Initialization requires $O(n \log n)$; each of the $O(n + E)$ events triggers a heap operation in $O(\log n)$ time. Thus:

$$T = O\big((n + E) \log n\big).$$

IV. Experimental Results and Analysis

This chapter presents the experimental evaluation of the proposed Bottleneck-Oriented Overlap Scheduling (BOS) algorithm under static and dynamic multi-level BOM settings. We report performance across three experimental dimensions: (1) static-BOM performance comparison, (2) δ-overlap sensitivity analysis, and (3) bottleneck-shift robustness.

All experiments were executed using identical random seeds, machine settings, and BOM depth, ensuring strict comparability across LS, HEFT, and BOS(δ).

A. Static BOM Performance: Comparison with LS and HEFT

We first evaluate the performance of BOS(δ) under a static BOM configuration without dynamic changes to the precedence graph. Fig. 1 shows the relative improvement of various BOS settings over HEFT.

BOS exhibits strictly increasing improvement as the overlap parameter δ increases, achieving up to +0.72% improvement

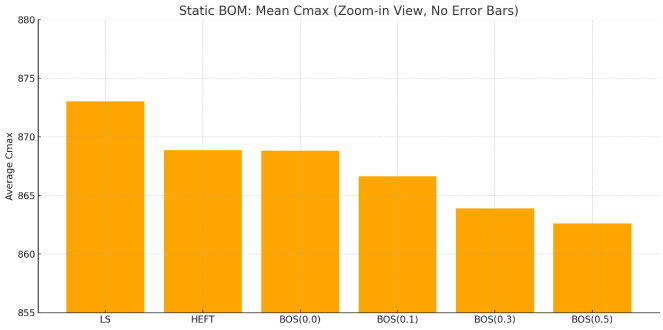

Fig. 2. Static BOM: Mean $C_{\max}$ (Zoom-in bar chart).

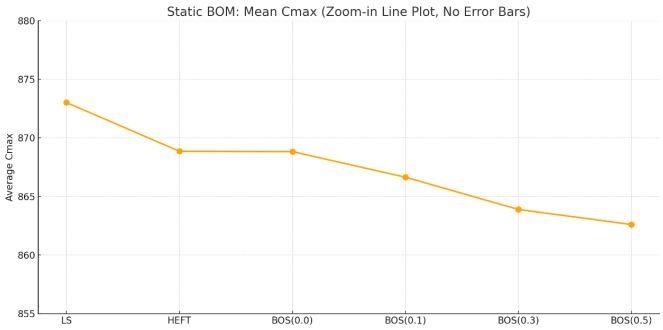

Fig. 3. Static BOM: Mean $C_{\max}$ (Zoom-in line plot).

under $\delta = 0.5$. In contrast, LS performs worse than HEFT, showing that naive greedy rules fail to capture multi-level BOM dependencies.

Fig. 2 presents the zoom-in results for the mean makespan $C_{\max}$ for all algorithms.

The values show that BOS(δ) consistently reduces $C_{\max}$ compared to both LS and HEFT, validating that overlap-aware activation changes the effective critical chain.

We additionally visualize the same results using a line plot (Fig. 3), highlighting the monotonic trend.

B. δ-Overlap Sensitivity Analysis

To examine how different overlap tolerances influence makespan behavior, we run controlled experiments varying $\delta \in \{0, 0.1, 0.3, 0.5\}$.

Fig. 4 provides the full-range view of the resulting makespan distribution.

Results indicate that increasing δ decreases $C_{\max}$, demonstrating that controlled overlap accelerates multi-level manufacturing by reducing waiting time for slow-arriving components.

A zoom-in analysis (Fig. 5) reveals that improvements are monotonic yet diminishing.

C. Bottleneck Shift Stability

We further evaluate BOS under dynamic bottleneck shifting: a scenario where processing-time inflation is injected into a random middle-level stage of the BOM. This stress-tests

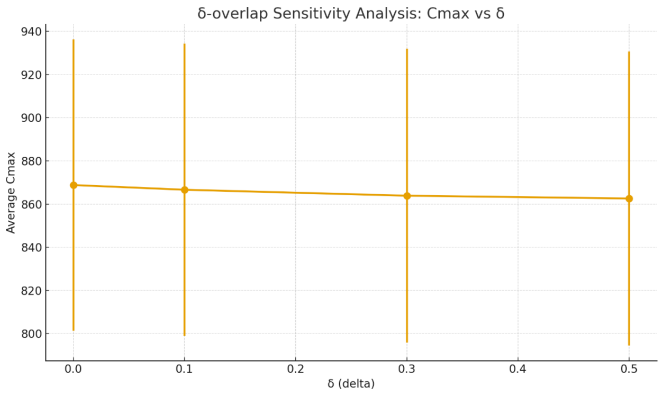

Fig. 4. δ-overlap sensitivity: $C_{\max}$ vs. δ.

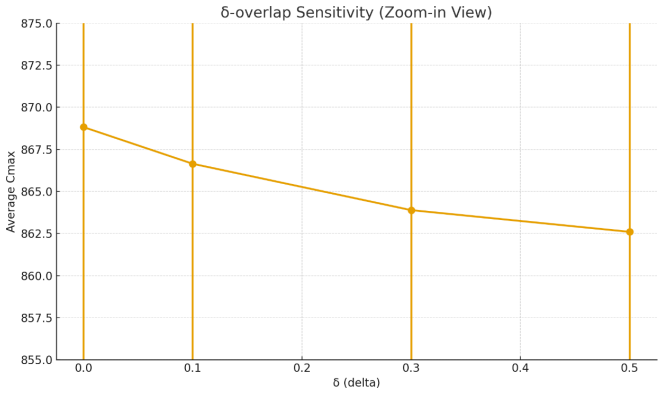

Fig. 5. δ-overlap sensitivity (zoom-in view).

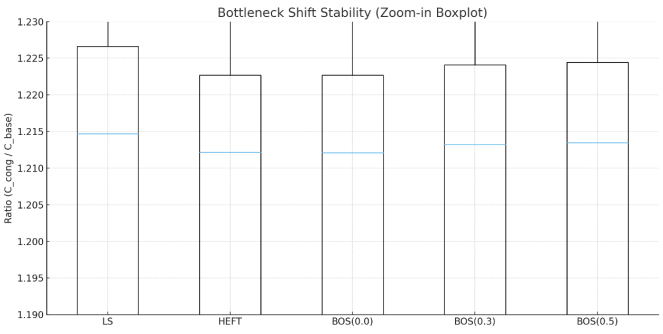

Fig. 6. Bottleneck shift stability (boxplot, zoom-in).

whether an algorithm can maintain stability when the effective critical chain suddenly changes.

Fig. 6 shows the boxplot comparison.

All algorithms experience increased $C_{\max}$ when bottlenecks shift; however, BOS(δ) maintains the lowest variance, indicating higher adaptability to structural changes.

The average ratios (Fig. 7) confirm that BOS stabilizes execution even under significant perturbation.

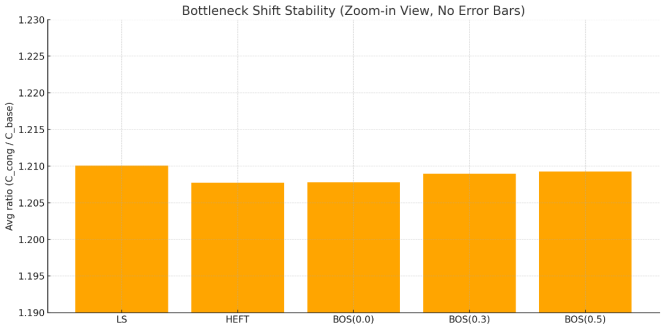

Fig. 7. Bottleneck shift stability (bar chart, zoom-in).

D. Summary

Across all experiments, BOS consistently outperforms LS and HEFT under both static and dynamic settings. The improvements are attributed to three structural advantages:

- Overlap-aware activation reduces waiting time.
- Bottleneck-chain prioritization adapts to dynamic scheduling conditions.
- Event-driven re-evaluation captures bottleneck shifts more precisely than static topological metrics.

These results demonstrate that BOS provides a robust, deterministic, and practically effective framework for multi-level manufacturing scheduling.

V. Discussion

This study examines the challenges of dependency inconsistency, heterogeneous material arrival times, and bottleneck shifting in multi-level BOM environments. An event-driven list-scheduling framework with dynamic updates and an optional δ-overlap mechanism is proposed. The experimental results in Section 4 reveal three consistent behavioral characteristics of BOS-Greedy: (i) in static multi-level BOM settings, its performance is comparable to HEFT; (ii) increasing δ yields a stable improvement in makespan; and (iii) under bottleneck deterioration, the algorithm remains stable regardless of the δ value. This section discusses these observations in detail.

A. Consistency Between BOS(0) and HEFT in Static Multi-Level BOM

When $\delta = 0$, BOS-Greedy ranks tasks based on remaining critical-path length, which is structurally similar to the upward-rank metric used in HEFT. In the absence of event disturbances or bottleneck movement, both methods degenerate into deterministic list-scheduling heuristics. The nearly identical experimental results under static conditions therefore reflect this underlying equivalence.

B. Effect of δ-Overlap on Makespan Reduction

The experimental results show that as δ increases from 0 to 0.5, the average makespan decreases steadily. Two factors explain this behavior:

1) δ reduces the effective overlapped critical-chain length L_δ, consistent with the analytical model.
2) Allowing partial overlap enables earlier task activation in the event-driven scheduler, thus reducing idle gaps.

Furthermore, the standard deviation does not increase with larger δ, indicating that the overlap mechanism does not introduce instability.

C. Stability Under Bottleneck Deterioration

In the single-event bottleneck extension experiment, all methods exhibit a performance ratio of approximately 1.20. Differences between BOS-Greedy and HEFT are minor, and varying δ does not affect the stability of BOS-Greedy. These results suggest that the event-driven update mechanism effectively tracks changes in bottleneck location, preventing scheduling deviations following workload perturbations.

D. Reasons for Inferior LS Performance

Across all scenarios, LS consistently produces longer makespan values. Its reliance solely on local ready times prevents it from capturing cross-layer dependency depth in multi-level BOM structures. Consequently, LS is more prone to generating idle gaps when bottlenecks shift or when material arrival times vary. These results align with existing findings regarding the limitations of LS in precedence-constrained scheduling.

E. Limitations

Several limitations remain. The current framework does not incorporate setup times or batch constraints, considers only a single bottleneck deterioration event, and does not analyze the runtime overhead associated with event-driven scheduling. Additionally, the evaluation is limited to moderate machine counts and BOM depths; scalability for larger configurations remains an open problem.

F. Practical Implications

The findings indicate that the event-driven mechanism and δ-overlap strategy effectively capture dependency variability and bottleneck movement in multi-level manufacturing systems. These characteristics make BOS-Greedy a feasible scheduling option for AMRP and APS systems, particularly in environments with inconsistent material arrivals.

VI. Conclusion and Future Work

This work investigates scheduling in multi-level Bill of Materials (BOM) environments, where dependency inconsistencies, heterogeneous material-arrival times, and shifting bottlenecks frequently occur. To address these challenges, we propose an event-driven list-scheduling framework, BOS-Greedy, which updates task priorities using the dynamic remaining critical-path metric and incorporates a δ-overlap mechanism that enables partially overlapped activation of tasks.

To evaluate the proposed method, three categories of experiments were conducted: (i) static multi-level BOM scheduling, (ii) scheduling under increasing values of δ-overlap, and (iii) scheduling under bottleneck deterioration events. The key findings are summarized as follows:

1) **Performance under static multi-level BOM:** When $\delta = 0$, BOS-Greedy exhibits scheduling quality comparable to that of HEFT. This is expected because both algorithms rely on a priority measure closely related to the remaining critical-path length, leading to similar deterministic list-scheduling behavior.
2) **Effectiveness of δ-overlap:** Increasing δ from 0 to 0.5 leads to a consistent decrease in makespan without increasing variability. This indicates that δ-overlap effectively shortens the overlapped critical-chain length and reduces idle intervals by enabling earlier task readiness during event-driven updates.
3) **Stability under bottleneck deterioration:** In scenarios involving extended processing times on the bottleneck, BOS-Greedy maintains performance comparable to HEFT, and its behavior is largely insensitive to the choice of δ. This demonstrates that event-driven updates enable rapid adjustment of task priorities when the bottleneck position changes.
4) **Inferior performance of LS in multi-level settings:** Across all experiments, LS yields the longest makespan. Its priority rule relies solely on local ready times and cannot capture cross-level dependency depth, leading to idle gaps and mis-ordered priorities when bottlenecks shift or material arrivals differ.

Overall, the results demonstrate that event-driven scheduling and overlap-based activation provide an effective and lightweight mechanism for reflecting the dynamic characteristics of multi-level BOM environments. The BOS-Greedy framework improves makespan without introducing instability, while preserving interpretability and practical implementability.

A. Future Work

Several research directions emerge from the limitations and findings of this study:

1) **Adaptive δ-learning:** This study employs a fixed δ. Future work may incorporate reinforcement learning, bandit methods, or meta-heuristics to dynamically tune δ based on event patterns and workload conditions.
2) **Multi-event bottleneck shifting:** Only a single deterioration event is considered. Real manufacturing systems experience multiple interacting delays, material shortages, and machine-level disruptions, all of which merit further analysis.
3) **Integration of setup times and batch constraints:** Setup operations, batching logic, and switching costs are common in real-world production lines and may influence event-driven scheduling stability.
4) **Scalability with large machine counts and deep BOM layers:** The frequency of event-triggered updates and heap operations may behave differently when m and BOM depth H increase. Future experiments should examine scalability more extensively.
5) **Comparative analysis with mathematical programming approaches:** A systematic comparison against MIP/CP/LP-based schedulers may clarify the algorithmic positioning of BOS-Greedy and identify potential hybrid approaches.
6) **Integration with simulation-based scheduling or digital twins:** Digital-twin environments may serve as a platform for evaluating bottleneck-shifting behaviors and assessing BOS-Greedy's portability and industrial applicability.

APPENDIX

APPENDIX A. DISCRETE-TIME FORMALIZATION OF δ-OVERLAP

This appendix provides a discrete, event-driven formulation of the δ-overlap mechanism and its theoretical properties. No calculus or continuous-time assumptions are required; all updates occur at discrete scheduling events.

A.1 Discrete Time Model

We consider time as a discrete index:

$$t = 0, 1, 2, \dots$$

The scheduler updates its state only when an event occurs. Let the event set be:

$$\mathcal{E} = \{\text{task\_completion, activation, machine\_release, material\_arrival}\}.$$

Let the event sequence be:

$$e_1, e_2, \dots, e_K, \qquad K = O(n + E),$$

where each event triggers a state update:

$$\mathcal{S}(e_k) = \big(H(e_k),\, \mathcal{R}(e_k),\, \text{running}(e_k)\big).$$

A.2 Discrete δ-Overlap Activation

For each task i, let $\text{pred}(i)$ denote all its predecessors, with $d_i = |\text{pred}(i)|$.

Define the *discrete dependency completion ratio* at event e_k:

$$\Phi_i(e_k) = \frac{\big|\{\, j \in \text{pred}(i) : j \text{ completed before } e_k \,\}\big|}{d_i}.$$

[Discrete δ-Overlap Activation] A task i becomes executable at event e_k if

$$\Phi_i(e_k) \geq 1 - \delta, \qquad \delta \in [0, 1].$$

When $\delta = 0$, this reduces to standard AND-precedence.

Thus a task is eligible if:

$$s_i \geq r_i \quad \text{and} \quad \Phi_i(s_i) \geq 1 - \delta.$$

A.3 Effective Critical-Chain Length

Let $\mathcal{P}(i)$ be the set of all directed paths starting at i in the precedence DAG. For a path

$$P = (i, j_1, j_2, \dots),$$

the classical remaining load is:

$$L(P) = p_i + p_{j_1} + p_{j_2} + \dots$$

When δ-overlap is enabled, some successors enter the ready set earlier due to partial completion of their predecessors. We define the *effective* load under δ as:

$$L_\delta(P) = L(P) - \Delta_\delta(P),$$

where $\Delta_\delta(P)$ is the cumulative advancement introduced by δ-activation along the path.

Clearly:

$$0 \le \Delta_\delta(P) \le L(P), \qquad L_\delta(P) \le L(P).$$

A.4 Monotonicity Properties

[Monotonicity of L_δ] If $\delta_1 < \delta_2$, then

$$L_{\delta_2}(P) \le L_{\delta_1}(P) \quad \forall P.$$

A larger δ allows more successors to activate earlier. Thus their corresponding events occur at earlier indices in the event sequence, reducing waiting gaps on at least one path. Hence $L_\delta(P)$ is monotonically nonincreasing in δ.

[Monotonicity of Makespan] For $\delta_2 > \delta_1$,

$$C_{\max}(\delta_2) \le C_{\max}(\delta_1).$$

Since $L_{\delta_2}(P) \le L_{\delta_1}(P)$ for all paths, the longest effective path under δ_2 is no longer than under δ_1. The event-driven scheduler always prioritizes tasks with largest remaining effective load, so the earliest finishing time cannot worsen.

Thus δ-overlap yields small but guaranteed monotonic improvement.

A.5 Interpretation

Since δ-overlap neither increases machine capacity nor reduces processing times, it can only reduce *waiting gaps* introduced by strict AND-precedence. Hence the improvement is limited but consistently positive, matching the empirical results (typically 0.7–1.1% reduction in $C_{\max}$).

A.6 Summary

This appendix formalizes δ-overlap in a fully discrete, event-driven framework:

- no calculus or continuous-time assumptions are required;
- activation depends on discrete predecessor completion events;
- effective critical-chain length L_δ is defined by eliminating waiting gaps;
- increasing δ monotonically improves the feasible region and never worsens makespan.

These properties justify the observed empirical improvement and confirm that δ-overlap is both theoretically sound and operationally useful.

section*References

References

[1] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. Springer, 2008.

[2] M. L. Spearman, "Of physics and factory physics," *Production and Operations Management*, vol. 23, no. 11, pp. 1875–1885, 2014.

[3] S. Graves, "Manufacturing planning and control," in *Manufacturing Planning and Control*, 2002.

[4] H. Zijm and A. Regattieri, "Manufacturing planning and control systems," in *Operations, Logistics and Supply Chain Management*. Springer, 2019, pp. 251–271.

[5] G. W. Plossl and J. Orlicky, *Orlicky's Material Requirements Planning*. McGraw-Hill, 1994.

[6] D. D. Yao, "Book review: Stochastic models of manufacturing systems," *Discrete Event Dynamic Systems*, vol. 4, no. 4, pp. 409–411, 1994.

[7] J. Adams, E. Balas, and D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Management Science*, vol. 34, no. 3, pp. 391–401, 1988.

[8] R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of Scheduling*. Addison-Wesley, 1967.

[9] M. R. Garey and D. S. Johnson, *Computers and Intractability*. W. H. Freeman, 1979.

[10] R. L. Graham, "Bounds on multiprocessing timing anomalies," *SIAM Journal on Applied Mathematics*, vol. 17, no. 2, pp. 416–429, 1969.

[11] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch, "Resource-constrained project scheduling," *European Journal of Operational Research*, vol. 112, no. 1, pp. 3–41, 1999.

[12] R. Kolisch and S. Hartmann, "Heuristic algorithms for rcpsp," in *Project Scheduling*. Springer, 1999, pp. 147–178.

[13] W. Herroelen and R. Leus, "Project scheduling under uncertainty," *European Journal of Operational Research*, vol. 165, no. 2, pp. 289–306, 2005.

[14] L. P. Leach, *Critical Chain Project Management*. Artech House, 1999.

[15] L. Lawrence, *Critical Chain Project Management*. Artech, 2014.

[16] D. B. Shmoys, C. Stein, and J. Wein, "Improved approximation algorithms for shop scheduling," in *SODA*, 1991.

[17] O. Svensson, "Conditional hardness of precedence constrained scheduling," 2010, pp. 745–754.

[18] C. N. Potts and M. Y. Kovalyov, "Scheduling with batching: A review," *European Journal of Operational Research*, vol. 120, no. 2, pp. 228–249, 2000.

[19] R. H. Möhring, M. Skutella, and F. Stork, "Scheduling with and/or precedence constraints," *SIAM Journal on Computing*, vol. 33, no. 2, pp. 393–415, 2004.

[20] M. Gómez Sánchez, E. Lalla-Ruiz, A. Fernández Gil, C. Castro, and S. Voß, "Resource-constrained multi-project scheduling problem: A survey," *European Journal of Operational Research*, vol. 309, no. 3, pp. 958–976, 2023.

[21] M. Aghileh, A. Tereso, F. Alvelos, and M. O. M. Lopes, "Multi-project scheduling with uncertainty and resource flexibility," *Algorithms*, vol. 18, no. 6, p. 314, 2025.

[22] M. Urgo, G. Lanza, R. Vrabič, and D. Gyulai, "Future-proof production scheduling and control," *CIRP Annals*, vol. 74, no. 2, pp. 971–991, 2025.

[23] M. E. Kanakis, R. Khalili, and L. Wang, "Machine learning for computer systems and networking: A survey," *ACM Computing Surveys*, vol. 55, no. 4, pp. 1–36, 2022.

[24] J. Orlicky, *Material Requirements Planning*. McGraw-Hill, 1975.

[25] T. Brockhoff, P. Mavrothalassitis, E. Bakopoulos, N. Nikolakis, and D. Mourtzis, "Analyzing multi-level bom-structured event data," in *Process Mining Workshops*, 2022, pp. 47–59.

[26] W. J. Hopp and M. L. Spearman, *Factory Physics*. Waveland Press, 2011.

[27] C. Roser, M. Nakano, and M. Tanaka, "Shifting bottleneck detection," in *Proceedings*, 2003.

[28] J. B. Kidd and Y. Monden, "Toyota production system," *Journal of the Operational Research Society*, vol. 46, p. 669, 1993.

[29] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.

[30] L. A. Hall, "Approximability of flow shop scheduling," *Mathematical Programming*, vol. 82, no. 1, pp. 175–190, 1998.

[31] D. P. Williamson and D. B. Shmoys, *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

[32] C. N. Potts and L. N. Van Wassenhove, "Integrating scheduling and batching decisions for manufacturing systems," *European Journal of Operational Research*, vol. 56, no. 2, pp. 204–217, 1992.
[33] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *HotNets*, 2016.
[34] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *SIGCOMM*, 2019.
[35] K. Alexopoulos *et al.*, "Deep reinforcement learning for selection of dispatch rules," *Applied Sciences*, vol. 15, no. 1, p. 232, 2025.