Response to the Referee Report

on the PhD dissertation of Krisztián Attila Bakon
entitled “Solving Industry 4.0 scheduling tasks”
Reviewer: Dr. Olivér Hornyák
Date: 23 April 2026

I would like to thank Dr. Olivér Hornyák for the thorough and constructive review of my dissertation and for the positive evaluation of the research topic, methodology, scientific contributions, and practical relevance of the work. I also appreciate the critical remarks and questions, as they provide an opportunity to clarify the intended interpretation of several modelling choices, the computational complexity of the proposed methods, and the limitations of the S-graph-based scheduling framework.

In the following, I first respond to the critical remarks listed in the review, and then answer the three questions raised by the reviewer.

Response to the critical remarks

Remark 1 — Page 31, formulation of the objective

Reviewer’s remark:
“Among all feasible schedules satisfying the above constraints, S_{n+1} should minimize the makespan” should have been mathematically formalized.

Response:
I thank the reviewer for pointing this out. The sentence was intended as a compact textual formulation of the standard makespan minimization objective in the reactive scheduling problem. Using the notation of the dissertation, the intended meaning is that the updated schedule S_{k+1} is selected from the set of feasible schedules satisfying the stated constraints and minimizes the maximum completion time.

More formally, let $\mathcal{F}_{k+1}$ denote the set of feasible updated schedules that include the newly arrived order, preserve the already started part of the previous schedule, and satisfy the release-time, precedence, and resource constraints. Then the intended objective can be written as

$$S_{k+1} \in \arg\min_{S \in \mathcal{F}_{k+1}} C_{max}(S),$$

were

$$C_{max}(S) = \max_{\left(i,j,t_i^s,t_i^f\right) \in S} t_i^f .$$

Thus, S_{k+1} denotes the selected feasible updated schedule with minimum makespan. The dissertation expresses this objective in prose at this point, while the algorithmic

sections apply this intended makespan-minimizing interpretation throughout the reactive scheduling framework.

Remark 2 — Page 33, Equation (3.3), common weighting coefficients

Reviewer's remark:
Equation (3.3) uses β as a common weight coefficient for earliness and tardiness penalties. Equation (3.1) had separated, product-specific weights, w_i^E and w_i^T, respectively. From a modeling perspective, the latter formulation is more general and more expressive, since it allows different products to have different penalty structures for early or late completion. For this reason, the product-specific weighting scheme would have been preferable in the later formulation as well.

Response:
I agree with the reviewer that the formulation with product-specific earliness and tardiness weights is the more general one. The use of the common coefficient β in Equation (3.3), however, was intended to serve a different modelling purpose. In that equation, β is not meant to replace product-specific penalty structures; rather, it acts as a scalarization coefficient controlling the relative importance of the aggregated due-date-related term against the intermediate storage term.

In other words, Equation (3.3) should be interpreted as a higher-level trade-off between two objective groups:

$$IS_{\text{total}}$$

and

$$E_{\text{total}} + T_{\text{total}}.$$

The formulation used in the dissertation corresponds to the special case where all products are treated with equal earliness and tardiness importance within the aggregated due-date term. This choice was made to keep the comparison of the storage-related and due-date-related objectives transparent and to focus the methodological discussion on the integration of intermediate storage into the S-graph-based framework.

The more general weighted version is fully compatible with the proposed method and can be written as

$$\min\left(\alpha \cdot IS_{\text{total}} + \beta \cdot \sum_{p \in P}\left(w_p^E E_p + w_p^T T_p\right)\right).$$

This form preserves the role of α and β as weights between the main objective components, while w_p^E and w_p^T express product-specific priorities. The branch-and-bound structure, feasibility checks, S-graph representation, and timing logic remain unchanged under this more general objective; only the numerical evaluation of candidate schedules is modified. Therefore, the common-weight formulation in the

dissertation should be understood as a simplified and transparent special case of the more general product-weighted model.

Remark 3 — Page 42, global node notation

Reviewer's remark:
"Add zero-wait arcs from global node Z." This chapter referenced global node as n_0. This is a minor notation inconsistency.

Response:
I thank the reviewer for identifying this notational issue. The symbols Z and n_0 refer to the same conceptual object in the S-graph construction: the artificial global source node used as the origin of release-time and zero-wait constraints.

In the dissertation, n_0 is introduced as the formal notation for the global source node. The later use of Z was intended as a mnemonic notation emphasizing the role of this node as a zero-time or zero-wait anchoring node in the initialization of the reactive S-graph. No additional node or different modelling construct is intended by Z. Thus, the intended interpretation is

$$Z \equiv n_0.$$

This notation does not affect the graph structure, the feasibility conditions, or the operation of the algorithm; it only concerns the naming of the same artificial source node.

Remark 4 — Page 48, Figure 4.8(A), length of task 9:1

Reviewer's remark:
In Figure 4.8(A), the length of task 9:1 is different from that of (B), (C), and (D).

Response:
I thank the reviewer for the careful observation. The different length of task 9:1 in Figure 4.8(A) is a consequence of the machine-dependent processing times in the flexible job shop model. In subfigure (A), task 9:1 is assigned to machine J_2, whereas in subfigures (B), (C), and (D) it is assigned to machine J_3. Since the processing time of a task depends on the selected equipment, the same task may have different durations on different machines.

Formally, the processing time should be interpreted as $p_{i,j}$, where i denotes the task and j denotes the assigned machine. Hence, for task 9:1,

$$p_{9:1,J_2} \neq p_{9:1,J_3}$$

may hold. Therefore, the different bar length does not represent a graphical error, but reflects a different machine assignment under the compared rescheduling policies.

Remark 5 — Page 59, internal storage reduction algorithm and objective function (5.8)

Reviewer's remark:
The internal storage reduction algorithm explained in the first paragraph may break objective function (5.8).

Response:
I accept the reviewer's concern, and I would like to clarify the intended role of the internal storage reduction step. The storage reduction procedure is not intended to override the full objective function or to act as an unconditional post-processing operation. Its intended role is to perform a local timing adjustment that attempts to reduce unnecessary intermediate storage while preserving schedule feasibility and while remaining consistent with the selected objective function.

The objective function combines intermediate storage and due-date-related terms. Therefore, a local reduction of IS_{total} is beneficial only if it does not lead to a deterioration of the complete weighted objective, unless the chosen weights explicitly prioritize storage reduction over due-date performance. In this sense, the local storage-reduction step should be interpreted as an objective-aware improvement mechanism.

More precisely, if S denotes the current candidate schedule and S' denotes the schedule obtained after a local storage-reduction adjustment, then the modified schedule is preferable only if

$$F(S') \leq F(S),$$

where

$$F(S) = \alpha \cdot IS_{\text{total}}(S) + \beta \cdot \left(E_{\text{total}}(S) + T_{\text{total}}(S)\right).$$

Thus, the local adjustment is not meant to minimize storage independently from the due-date objective. It is meant to reduce storage slack where this is feasible and beneficial with respect to the complete scalarized objective. The branch-and-bound search still evaluates candidate schedules according to the full objective function, and the storage-reduction logic is therefore subordinate to the overall optimization criterion.

Answers to the reviewer's questions

Question 1

Question:
The method proposed in this document is based on a branch-and-bound search over S-graph representations. Given that flexible job shop scheduling is an NP-hard problem, how does the worst-case computational complexity of your algorithm scale with the number of tasks and machines? Additionally, can you identify the dominant factors that most significantly influence the growth of the search space in the presented approach?

Answer:
The proposed S-graph-based branch-and-bound method has exponential or factorial worst-case complexity, as expected for an exact method applied to a flexible job shop scheduling problem. The branch-and-bound procedure improves practical solvability through lower bounds, dominance rules, feasibility checks, and pruning, but it does not change the worst-case complexity class of the underlying problem.

Let n denote the number of tasks to be scheduled. In the reactive case, n refers to the number of tasks that remain free for rescheduling after the already started or already executed part of the schedule has been fixed. Let m denote the number of candidate machines. In a fully flexible worst-case setting, where each task can be processed on each machine and precedence restrictions are ignored for the purpose of estimating the size of the search space, inserting a task into one of the existing machine sequences gives approximately $m + k$ possible insertion positions at depth k, where k tasks have already been scheduled. Thus, a representative upper-bound characterization of the number of possible leaf schedules is

$$\prod_{\boldsymbol{k=0}}^{\boldsymbol{n-1}}(\boldsymbol{m}+\boldsymbol{k})\,\frac{(n+m-1)!}{(m-1)!}.$$

This expression is factorial in n for fixed m, and remains super-polynomial when both the number of tasks and the degree of routing flexibility increase. The total computational effort also includes the per-node cost of feasibility testing, graph updates, timing propagation, and lower-bound computation; these operations add polynomial overhead to the combinatorial growth of the search tree.

The dominant factors influencing the growth of the search space are the number of unscheduled or reschedulable tasks, the number of eligible machines per task, the number of feasible insertion positions on each machine, the density of precedence and storage-related constraints, and the effectiveness of the lower bounds and pruning rules. In the reactive framework, fixing the already started part of the schedule reduces the practical size of the search problem, but the remaining subproblem is still exponential in the worst case.

Question 2

Question:
The introduced policy-based restrictions, e.g. append-only, partial insertion, full rescheduling, effectively constrain the search space. How do these policies influence the theoretical and practical complexity of the algorithm? In particular, can you characterize whether these restrictions reduce the problem size in a provable way, or do they primarily act as heuristics without guarantees on complexity reduction?

Answer:
The policy-based restrictions should be viewed as formal constraints on the admissible rescheduling domain, not merely as informal heuristics. Append-only,

partial insertion, and full rescheduling define different levels of rescheduling freedom. A stricter policy fixes a larger part of the existing schedule and therefore admits only a subset of the schedules that would be feasible under a more permissive policy.

For a fixed instance, this gives a clear monotonic relation between the policies: the admissible search space under a stricter policy cannot be larger than the admissible search space under a less restrictive one. For example, append-only preserves the existing schedule and only allows the new tasks to be placed after the relevant fixed part of the schedule. Partial insertion increases the admissible set by allowing new tasks to be inserted into existing gaps or by allowing limited temporal shifts. Full rescheduling gives the largest feasible domain among the considered policies because all not-yet-started tasks may be reassigned and resequenced.

However, these restrictions do not change the worst-case complexity class of the method. After applying a policy, the remaining problem is still solved by exact branch-and-bound over S-graph-based assignment and sequencing decisions. Therefore, the residual search remains exponential or factorial in the number of tasks that remain free for rescheduling.

From a practical point of view, the policies are nevertheless highly significant. Append-only is typically the least expensive computationally because it fixes almost the entire existing schedule. Partial insertion reduces the branching factor by restricting the admissible insertion positions and preserving part of the original sequence. Full rescheduling is computationally the most expensive, because it leaves the largest number of decisions open. Thus, the policies provide a provable reduction of the feasible decision space for a fixed instance, but they do not eliminate the NP-hard nature of the residual scheduling problem. Their main role is to provide a controllable trade-off between schedule stability, optimization freedom, and computational effort.

Question 3

Question:
What are the limitations of applying the S-graph framework to more general or heterogeneous production systems, such as those with sequence-dependent setup times or transportation delays?

Answer:
The main limitation of applying the proposed S-graph framework to more general or heterogeneous production systems is that its natural representation is based on tasks, recipe arcs, equipment sequencing arcs, and temporal constraints derived from precedence, machine assignment, release times, and storage policies. This structure is well suited to the flexible job shop scheduling problems considered in the dissertation, especially in the presence of equipment flexibility, due dates, dynamic order arrivals, and NIS/UIS intermediate storage assumptions. However, richer heterogeneous production environments may contain additional state-dependent or

multi-resource interactions that are not represented directly by the basic S-graph structure.

Sequence-dependent setup times can be incorporated naturally when the setup depends only on two consecutive operations assigned to the same machine. In such a case, the setup can be represented through modified schedule-arc weights or additional sequencing logic. However, more complex setup structures are more difficult to model directly. For example, setups may depend on product families, machine states, cleaning requirements, batch composition, previous processing history, or the availability of a shared setup crew. These cases require additional state variables, auxiliary tasks, dummy nodes, or problem-specific graph transformations.

A similar limitation applies to transportation delays. If transportation can be approximated as a waiting time, blocking relation, release constraint, or storage-related delay, then it can be represented within an extended S-graph formulation. However, explicit transportation systems involving routing-dependent travel times, limited vehicles, AGV conflicts, robot coordination, finite buffer capacities, or time-dependent travel behaviour introduce additional resource-allocation decisions. These decisions are not naturally captured by recipe arcs and machine sequencing arcs alone.

Therefore, the limitation is not that the S-graph framework cannot be extended, but that such extensions increase modelling and computational complexity. Additional heterogeneous features generally require extra nodes, arcs, resources, synchronization constraints, or state-dependent logic. This makes the model more problem-specific and increases the size of the branch-and-bound search space. Consequently, the proposed S-graph approach is most natural for structured flexible job shop environments, while richer heterogeneous production systems require nontrivial model extensions and may face stronger scalability limitations.

Closing statement

I hope that the above responses clarify the issues raised in the review. I am grateful for the reviewer's constructive comments and questions, which provided an opportunity to explain more precisely the intended interpretation of the mathematical formulation, the role of the policy-based restrictions, the computational complexity of the exact S-graph-based branch-and-bound procedure, and the natural modelling limits of the proposed framework.

Nagykanizsa, 23 April 2026

Krisztián Attila Bakon