Annals of Operations Research (2026) 358:1169–1198
https://doi.org/10.1007/s10479-026-07087-z

ORIGINAL RESEARCH

Logic-based Benders decomposition for additive manufacturing scheduling on unrelated parallel machines

Jian Chen[1] · Wenjing Ma[1] · Mingyue Sun[2] · Zhiheng Zhao[2]

Received: 28 October 2024 / Accepted: 26 January 2026 / Published online: 16 February 2026

Abstract
Additive manufacturing (AM), also known as 3D printing, offers innovative solutions for customized production. This paper addresses an AM scheduling problem on unrelated parallel machines that considers different part release times. A mixed integer linear programming (MILP) model aimed at minimizing makespan is developed, enhanced by valid inequalities and symmetry-breaking constraints. We propose a logic-based Benders decomposition (LBBD) algorithm to divide the problem into a master problem (MP), which determines part assignment and grouping into batches, and multiple subproblems (SPs) that sequence the batches on each AM machine. An earliest release date (ERD) rule is introduced to quickly solve subproblems. Three feasibility cuts and two optimality cuts are proposed and combined into eight types of cuts for the master problem. In addition, a new heuristic is developed to quickly obtain an upper bound of the problem. Computational results across various instance sizes demonstrate the performance of the proposed LBBD algorithm, as well as the effectiveness of the proposed symmetry-breaking constraints and various types of cuts.

Keywords Scheduling · Logic-based Benders decomposition · Additive manufacturing · Unrelated parallel machines

1 Introduction

Additive manufacturing (AM), which evolved from rapid prototyping techniques, is commonly known as 3D printing. Unlike traditional formative or subtractive manufacturing, AM employs a 'layer-by-layer' approach creating objects directly from 3D model data, enabling the precise construction of complex, composite and hybrid structures with remarkable design flexibility (Tofail et al., 2018). This unique capability gives AM the potential to spearhead a

✉ Mingyue Sun
ming-yue.sun@connect.polyu.hk

Jian Chen
jchen@nuaa.edu.cn

Wenjing Ma
wenjingm@nuaa.edu.cn

[1] College of Economics and Management, Nanjing University of Aeronautics and Astronautics, Nanjing, China

[2] Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hong Kong, China

new industrial revolution across various domains, including aeronautics, automotive, healthcare and defense (Kanishka and Acherjee, 2023). Among AM processes, selective laser melting (SLM)–a powder bed fusion (PBF) technology–stands out for producing near-full-density parts with fine resolution, often outperforming traditional forming and machining processes in quality and geometric freedom (Bahnini et al., 2018).

This paper studies AM scheduling problem for SLM process on unrelated parallel machines. In practice, an SLM machine operates as a batch processing machine: multiple parts can be fabricated simultaneously if they fit within the build chamber, yet no part can be added or removed once the production starts. The makespan of a batch arises from three coupled components: (i) a setup time shared by all parts, (ii) laser melting time that scales with each part's material volume, and (iii) powder-layering time determined by the tallest part (i.e., the total number of layers). Consequently, the total production time of a batch cannot be known until its composition is fixed, and efficient planning must jointly determine part grouping and batch sequencing to minimize makespan and utilize capacity.

In operational terms, running an AM batch typically involves three stages: preparation, production, and collection (Li et al., 2019). Preparation includes processing digital model data, preparing powder, establishing the protective atmosphere, and warming up the machine. During production, thin powder layers are spread over a base plate or previously consolidated material, and each layer is selectively melted by a high-power laser; layering and melting alternate until all parts in the batch are completed. The laser time closely follows part volume, while setup time is shared across parts (Li et al., 2017); the layering time depends on the height of the tallest part because it fixes the total number of layers. These intertwined drivers make the grouping–sequencing problem combinatorial and non-trivial, motivating scheduling methods that explicitly capture shared setup, volume-dependent processing, and height-coupled layering.

This paper investigates the unrelated AM machine scheduling problem with different release times (UAMSP-R), where parts can be produced simultaneously in batches. In industrial SLM lines, parts seldom become printable at time zero. Engineering changes, design verification, build-orientation/support generation, printability checks, material/plate availability, quality rework, and rush orders lead to staggered readiness times. We capture this reality by assigning each part a release date, the earliest time at which it can be batched and processed. Compared with the traditional batch scheduling problem, the processing time of a batch is a function of both the properties of the parts assigned to each batch and the attributes of the assigned AM machine, making the scheduling process more complex. This complexity requires deciding how to effectively group parts into batches, assign these batches to AM machines, and sequence them on each machine to minimize the overall makespan. To address this problem, most studies employ metaheuristic methods for AM scheduling (e.g., genetic algorithm (Kapadia et al., 2022), adaptive large neighborhood search (Hu et al., 2022), and iterated greedy (Ying et al., 2022)). While heuristics are attractive for large-scale, complex settings, exact methods remain indispensable for obtaining optimal solutions and establishing performance benchmarks. Therefore, the purpose of this study is to develop an exact method for the AM scheduling problem. This study makes the following key contributions:

- We first formulate a generic mixed-integer linear programming (MILP) model that minimizes the makespan of the UAMSP-R, and enhance it with valid inequalities and four types of symmetry-breaking constraints to strengthen the model.
- Building upon this formulation, we design an exact solution approach based on a logic-based Benders decomposition (LBBD) framework, which decomposes the original problem into a master problem (MP) and multiple subproblems (SPs). The MP

determines part grouping into batches and assigns these batches to machines, while the SPs sequence the batches assigned to each AM machine. Three feasibility cuts and two optimality cuts are proposed and combined into eight types of cuts for the master problem.
- Furthermore, an earliest-release-date (ERD) rule is incorporated into the SPs, and a constructive heuristic is introduced to provide an initial upper bound, both designed to improve computational efficiency.
- Finally, extensive numerical experiments are conducted to evaluate the effectiveness and scalability of the proposed model and algorithm.

The paper is organized as follows. Section 2 provides the literature review. Section 3 describes the problem in details, presents the mathematical formulation and makes some model improvements. In Section 4, we develop the LBBD algorithm, proposing several valid cuts, an ERD rule for SPs, and a heuristic for the initial solution. A comprehensive computational study is conducted in Section 5. The paper concludes in Section 6, together with some insights for future research directions.

2 Literature review

Section 2.1 reviews AM scheduling problems. Section 2.2 provides a perspective on the development of logic-based Benders decomposition algorithm.

2.1 Additive manufacturing scheduling problems

Research in AM traditionally focused on process-driven (Gardan, 2016) and design-driven (Thompson et al., 2016) aspects required to produce complex geometries (Oh et al., 2020). As the technology sector has matured, the demand for producing high volumes of customized parts using AM has grown significantly (Hedenstierna et al., 2019). This shift has led to an increasing emphasis on developing efficient scheduling methods for mass customization in AM to meet the diverse requirements of AM parts (Khorram Niaki and Nonino, 2017). From a scheduling perspective, however, research in AM remains limited.

The AM scheduling problem involves a series of intricate decisions aimed at optimizing AM production processes. Key tasks include grouping various parts into batches, assigning these batches to machines and sequencing batches on each machine. Additional considerations include capacity constraints, where certain parts may be unsuitable for specific machines due to factors like build area, height limitations, or precision requirements (Oh et al., 2020).

The AM scheduling problem shares similarities with BPM scheduling problem (Fowler and Mönch, 2022), where multiple parts can be processed simultaneously in a batch on a machine. Traditionally, two main types of batching are identified: serial batching (s-batch), where the batch processing time is determined by the sum of the processing times of the assigned parts, and parallel batching (p-batch), where the batch processing time is determined by the longest processing time among the parts within the batch. However, in the AM context, batch processing time depends on the physical characteristics of the included parts and the parameters of the AM machine, making it distinct from both s-batch and p-batch.

Li et al. (2017) first proposed a function for calculating the batch processing time on a SLM machine, which has since inspired further research. Ransikarbum et al. (2017) studied AM scheduling on non-identical parallel machines, aiming to minimize the cost, tardiness and unprinted builds while maximizing load balance. Fera et al. (2018) investigated a single-machine AM scheduling problem to minimize cost, weight total earliness and tardiness and

proposed a modified genetic algorithm for this purpose. Building on this, Fera et al. (2020) developed an improved Tabu search algorithm, which outperformed the earlier modified genetic algorithm in solving small to large instances. Kucukkoc (2019) made significant contributions to AM scheduling modeling by proposing three MILPs to address scheduling problems for single, identical parallel, and non-identical AM machines, with the goal of minimizing makespan. This paper uses a setup time calculation proposed by Kucukkoc. Yılmaz (2020) explored combining production and distribution in an AM supply chain to minimize the makespan and proposed a heuristic procedure with five selection rules. Based on a Q-Learning variable neighborhood search, Alicastro et al. (2021) proposed a reinforcement learning iterated local search meta-heuristic to provide good solutions in a low computational expense. Altekin and Bukchin (2022) addressed the production planning problem in multi-machine AM systems. A unified MILP minimizing the cost and makespan is formulated to solve the problem. Ying et al. (2022) extended the iterated greedy algorithm to solve a single-machine AM scheduling problem. Kapadia et al. (2022) addressed the problem of order acceptance and scheduling in AM on identical parallel machines with with the objective of maximizing total profit, and developed a random-keys based genetic algorithm tested on instances with up to 120 parts. Hu et al. (2022) considered AM scheduling on unrelated parallel machines with two-dimensional packing constraints, proposing an adaptive large neighborhood search for instances with up to 7 machines and 500 parts. Zipfel et al. (2024) addressed the customer order scheduling problem in additive manufacturing considering the parts family on unrelated parallel machines and proposed a metaheuristic based on an iterated local search with up to 120 parts. Besides, we review the literature on AM scheduling using exact algorithms. Mao et al. (2024) addressed a single machine scheduling problem in additive manufacturing considering two-dimensional constraints with the objective to minimize the makespan using combinatorial Benders decomposition, which also separates original problem into MP of allocating parts into build and multiple SPs of checking the packing feasibility without sequencing the builds. Nascimento et al. (2024) addressed a nesting and scheduling problem on identical parallel machines considering irregular-shaped parts aiming to minimize the tardiness, and proposed two LBBD algorithms. One combines MILP and constraint programming (CP), and the other relies solely on CP. It separates the problem into MP of allocating parts and sequencing builds, and multiple SPs of checking nesting feasibility.

Given the complexity of AM scheduling, most relevant studies have focused on heuristic algorithms, with fewer studies dedicated to exact algorithms. The existing studies on exact algorithms typically address single-machine or identical parallel machine scenarios. This paper aims to develop an exact algorithm for the AM scheduling problem, considering various release times on unrelated parallel machines. Given the two-stage structure of this problem where part grouping and assignment are handled separately from batch sequencing, logic-based Benders decomposition algorithm is identified as an effective solution approach.

2.2 Logic-based Benders decomposition algorithm

Benders decomposition (BD) was first proposed by Benders (1962) to tackle complex problems by temporarily fixing complicating variables, resulting in a significantly easier subproblem. The classical BD approach often fixes integer variables, transforming the problem into a continuous linear program where standard duality theory can be applied to develop cuts. Several extensions have broadened its application to a wider range of problems (Rahmaniani et al., 2017). However, when some optimization models include logic relations and rely

on big-M constraints, this transformation often leads to a weak formulation. Additionally, the subproblem cannot always be linearized, as it may include integer variables and nonlinear functions. Consequently, standard linear duality cannot be used to develop classical Benders cuts in such cases.

Hooker and Ottosson (2003) first introduced an extension known as logic-based Benders decomposition. LBBD is an advanced optimization technique that extends the classical Benders decomposition by integrating logic-based constraints into the decomposition framework. LBBD takes this a step further by allowing for more complex relationships and logical constraints to be incorporated into the SP. This approach is particularly useful for problems where traditional linear programming methods fall short, such as those involving integer variables or non-linear functions (Roshanaei et al., 2017). By effectively decomposing the problem into a MP and one or more SPs, LBBD generates cuts based on the logical structure of the problem, which helps in narrowing down the solution space. This approach is problem-specific and must be designed by fully exploiting the problem structure (Hooker, 2007). This method has found wide applications in various fields, including the scheduling problem (Chen et al., 2023; Guo et al., 2021; Zhang et al., 2022), the resource allocation optimization problem (Forbes et al., 2024), the humanitarian relief problem (Guo and Zhu, 2023), and the traveling repairman problem (Bruni et al., 2022), proving to be highly effective in handling large-scale and complex optimization problems.

Although, to the best of our knowledge, no study has yet applied LBBD to unrelated parallel AM machine scheduling. LBBD has proven effective for closely related parallel-machine scheduling settings (Fanjul-Peyro et al., 2019; Gedik et al., 2016; Naderi and Roshanaei, 2020). Recent advances extend this paradigm along several dimensions: Wu et al. (2024) addressed a bi-objective parallel-machine selection and scheduling problem with release dates and resource consumption, introducing an ϵ-constraint–based LBBD (ϵ-LBBD). To develop new LBBD-suitable mathematical models, Li et al. (2022) proposed three MILP formulations and a logic-based Benders algorithm for practical-sized instances, with a master that decides machine locations and job-to-machine assignments and subproblems that sequence jobs per machine. In a multi-factory context, Chen et al. (2024) incorporated order acceptance under carbon caps within an LBBD framework. Naderi and Roshanaei (2022) designed an efficient constraint programming (CP)–based LBBD method that leverages the complementary strengths of mixed-integer programming and CP for the flexible job shop scheduling problem. For distributed shop environments, Xiong et al. (2025) decomposed the distributed flow shop scheduling problem into an assignment master and multiple scheduling subproblems and proposed five enhanced LBBD approaches, while Xiong and Liu (2025) presented three LBBD frameworks that exploit decomposability to deliver optimal or near-optimal solutions with quantifiable quality guarantees under tight time limits for the distributed flexible job shop scheduling problem. Closer to our assignment–sequencing template, Wang et al. (2022) developed an LBBD approach for unrelated parallel machines with machine- and sequence-dependent setup times, using a batch-assignment master and a sequencing subproblem, together with LBBD-based heuristics that solve the master to controlled optimality gaps to balance speed and quality. However, none of these works considers the case where parallel machines are simultaneous batch processors, as in AM. In AM, parallel batch machines introduce the additional challenge of variable-size batching within a batch, which significantly increases modeling and algorithmic complexity and requires tailored decomposition and cut design beyond existing LBBD templates.

In conclusion, LBBD algorithm has emerged as a powerful extension of the classical Benders decomposition, addressing the limitations of traditional methods by incorporating logic-based constraints into the optimization framework. The adaptability of LBBD to a wide

range of applications, from resource allocation to complex machine scheduling problems, underscores its versatility. Despite challenges related to problem complexity and scalability, ongoing research continues to refine LBBD methodologies, enhancing computational strategies and expanding its applicability.

3 Problem formulation

In this section, we propose a formulation for UAMSP-R. We then introduce several formulation enhancement techniques, including valid inequalities and symmetry-breaking constraints. Lastly, we discuss the complexity of UAMSP-R.

3.1 Problem description and notations

The scheduling process of AM parts is illustrated in Figure 1. The AM service provider receives orders(parts) from customers with different release times. Taking into account production capacity and cost-effectiveness, parts from different customers may be grouped into a batch for production on AM machines.

Consider a set $I = \{1, 2, \ldots, |I|\}$ of AM machines and a set $R = \{1, 2, \ldots, |R|\}$ of parts to be produced. Each part $r \in R$ has a release time rt_r. In pursuit of capacity and resource efficiency, parts are organized into batches, represented as $B = \{1, 2, \ldots, |B|\}$ on each machine. Hence, any part produced within a batch cannot be removed until the entire batch is completed. Considering the characteristics of the AM machine, parts allocated to a batch on machine i cannot exceed the machine's maximum area capacity and height limit. Moreover, the batch processing time in this problem is represented as a function that considers the total volume and the maximum height of the allocated parts to be produced. To start a new batch on an AM machine, a series of operations are carried out to set up the machine, such as data preparation, powder material filling, machine adjustment, and protective atmosphere filling. Thus, the setup time is shared by all parts allocated to the same batch. In our paper, the proposed calculation method for the processing time of AM batches on machines PT_{ib} is adopted from Kucukkoc (2019). Once the parts are assigned to machines and grouped into

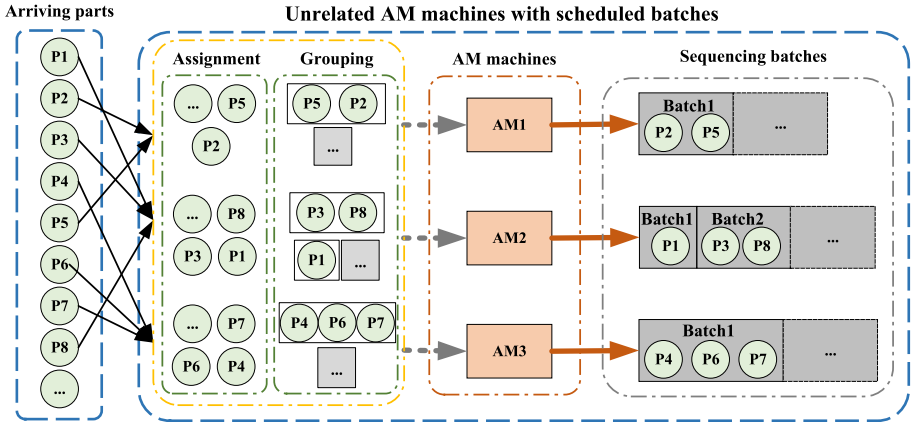

Fig. 1 The process of AM scheduling based on Kucukkoc (2019).

batches, the sequencing of batches on each machine needs to be determined. In this paper, we address part assignment and grouping decisions that incorporate batch sequencing on unrelated AM machines simultaneously.

We make several assumptions as follows:

(1) Each part must be executed in exactly one batch;

(2) An AM machine can produce more than one part simultaneously on its platform. Once the batch starts processing, no parts can be added to the batch. The production process cannot be interrupted;

(3) All machines are available at idle. An AM machine can only produce one batch at a time. The batches scheduled to a machine should be processed one by one in sequence.

The parts and machines have different specifications. The related parameters and variables of the problem are as follows:

Sets:

R, Set of parts, indexed by r

I, Set of machines, indexed by i

B, Set of batches, indexed by b,

Parameters:

rt_r, The release time of part r

h_r, The height of part r

v_r, The volume of part r

p_r, The minimum precision demand of part r

a_r, The area of part r

MH_i, The maximum height allowed for producing by machine i

MP_i, The printing precision of machine i

MA_i, The area of the machine i's tray

VT_i, Time spent to form per unit volume of material on machine i

HT_i, Time spent for powder-layering, which is repeated for each layer based on the highest part produced in the batch on machine i

SET_i, The setup time needed for initializing and cleaning of machine i

L, A large positive number

Decision variables:

x_{ibr}, Equal to 1 if part r is processed in batch b on machine i

v_{ib}, Equal to 1 if batch b on the machine i is not empty

$z_{ibb'}$, Equal to 1 if batch b is immediately processed before batch $b^{'}$ on machine i

H_{ib}, The height of batch b on machine i

PT_{ib}, The production time of batch b on machine i

C_{ib}, Completion time of batch b on machine i

$C_{\max}$, Makespan

3.2 Mathematical formulation

In scheduling theory, we often use three-field notation to describe problem characteristics. This problem is defined as $Rm\,|batch\,\{AM\}, rt_r, SET_i|\,C_{\max}$ problem. The objective is to minimize the makespan $C_{\max}$. We define a dummy batch 0 which represents the first and last batch processed on each AM machine. The original MILP model is built as follows:

$$\min C_{\max} \tag{1}$$

$$\sum_{i\in I}\sum_{b\in B} x_{ibr} = 1 \quad \forall r \in R \tag{2}$$

$$\sum_{r\in R} a_r x_{ibr} \le MA_i \quad \forall i \in I, b \in B \tag{3}$$

$$H_{ib} \ge h_r x_{ibr} \quad \forall i \in I, b \in B_i, r \in R \tag{4}$$

$$H_{ib} \le MH_i \quad \forall i \in I, b \in B \tag{5}$$

$$p_r x_{ibr} \le MP_i \quad \forall i \in I, b \in B, r \in R \tag{6}$$

$$PT_{ib} = SET_i v_{ib} + VT_i \sum_{r\in R}(v_r x_{ibr}) + HT_i H_{ib} \quad \forall i \in I, b \in B_i \tag{7}$$

$$\sum_{b'=0}^{B} z_{ibb'} = v_{ib} \quad \forall i \in I, b \in B \cup \{0\} \tag{8}$$

$$\sum_{b=0}^{B_i} z_{ibb'} = v_{ib'} \quad \forall i \in I, b' \in B \cup \{0\} \tag{9}$$

$$z_{ibb} = 0 \quad \forall i \in I, b \in B \tag{10}$$

$$C_{i0} = 0 \quad \forall i \in I \tag{11}$$

$$v_{i0} = 1 \quad \forall i \in I \tag{12}$$

$$C_{ib} \ge rt_r x_{ibr} + PT_{ib} \quad \forall i \in I, b \in B, r \in R \tag{13}$$

$$C_{ib} + PT_{ib'} - L(1 - z_{ibb'}) \le C_{ib'} \quad \forall i \in I, b \in B \cup \{0\}, b' \in B \tag{14}$$

$$C_{\max} \ge C_{ib} \quad \forall i \in I, b \in B \tag{15}$$

$$x_{ibr} \le v_{ib} \quad \forall i \in I, b \in B, r \in R \tag{16}$$

$$x_{ibr}, v_{ib}, z_{ibb'} \in \{0, 1\} \tag{17}$$

$$C_{\max}, C_{ib}, PT_{ib} \ge 0 \tag{18}$$

The objective function (1) minimizes the makespan. Constraints (2) impose that each part must be assigned to exactly one batch and one machine. Constraints (3) ensure the total area of parts assigned to a batch on machine i does not exceed area limit of the machine. Constraints (4) define the height of each batch on all machines. Constraints (5) and (6) make sure that the height and precision demand of parts allocated to a batch on machine i do not exceed the machine's maximum height and precision. Constraints (7) calculate the batch processing time on machine i depending on the total area of the allocated parts and the maximum height. Constraints (8) and (9) guarantee that each batch has exactly one predecessor and one successor batch. Constraints (10) mean each batch can only be processed once. Constraints (11) indicate that the completion time of the 0 batch (dummy) is 0. Constraints (12) indicate the dummy batch 0 is on every machine. Constraints (13) provide a valid lower bound for the completion time of each batch on each machine. They indicate that the completion time of batch b on machine i, if assigned with part r, is greater than or equal to its processing time PT_{ib} plus the release time rt_r. Constraints (14) guarantee that batches do not overlap on the same machine. Constraints (15) compute the makespan by enforcing it to be greater than or equal to the completion time of each batch. Constraints (16) indicate that a part cannot be assigned to a batch if the batch is empty. Constraints (17) and (18) bound the variables.

3.3 Formulation enhancement

3.3.1 Valid inequality

In this section, we introduce valid inequalities to strengthen the MILP model. We have the following inequalities:

$$C_{\max} \geq \sum_{b \in B} PT_{ib} v_{ib} + \min_{r \in R} rt_r \quad \forall i \in I \tag{19}$$

Inequalities (19) represent that, for all non-empty batches allocated to machine i, the makespan must be greater than or equal to the total processing time of all non-empty batches on each machine plus the minimum release time among all parts.

3.3.2 Symmetry-breaking constraints

We propose four symmetry-breaking constraints that eliminate symmetry scenarios within the solutions. These scenarios may lead to a large number of redundant solutions with the same objective value. Figure 2 presents the two types of symmetry scenarios we consider.

Figure 2(a) illustrates the symmetry between grouping interrupted by empty batches and consecutive batches without empty ones. The constraint SBC1 ensures that each non-empty batch on each machine is contiguous and numbered sequentially, where B' does not include the last batch on each machine.

$$\textbf{(SBC1)} \quad v_{i,b} \geq v_{i,b+1} \quad \forall i \in I, b \in B' \tag{20}$$

Then constraints SBC2, SBC3 and SBC4 are introduced to break the symmetry among the grouping solutions with the same objective value, which are independent of batch sequencing on a machine, see Figure 2(b), (c) and (d). The constraint SBC2 breaks the symmetry based

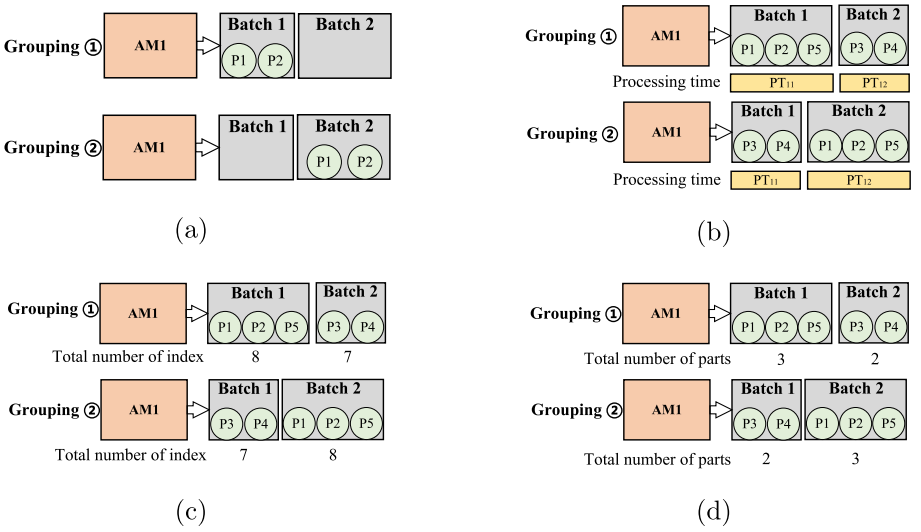

Fig. 2 Symmetry scenarios of part grouping on a machine.

on processing time of batches.

$$\textbf{(SBC2)} \quad PT_{ib} \geq PT_{ib+1} \quad \forall i \in I, b \in B' \tag{21}$$

The constraint SBC3 breaks the symmetry between batches based on the sum of part indices in the batch.

$$\textbf{(SBC3)} \quad \sum_{r \in R} r \cdot x_{i,b,r} \geq \sum_{r \in R} r \cdot x_{i,b+1,r} \quad \forall i \in I, b \in B' \tag{22}$$

The constraint SBC4 breaks the symmetry based on the total number of parts in the batch.

$$\textbf{(SBC4)} \quad \sum_{r \in R} x_{i,b,r} \geq \sum_{r \in R} x_{i,b+1,r} \quad \forall i \in I, b \in B' \tag{23}$$

Note that SBC1 removes the structural symmetry caused by inserting empty batches by enforcing that non-empty batches on each machine are contiguous. In contrast, SBC2-SBC4 address the same permutation symmetry among non-empty batches (independent of batch sequencing) by imposing a monotone ordering of batch indices using different ordering keys (batch processing time, weighted sum of part indices, or batch size). Therefore, SBC2-SBC4 should be regarded as alternative tie-breaking rules rather than additive constraints. Enforcing two (or more) of SBC2-SBC4 simultaneously may introduce conflicting ordering requirements, which can exclude feasible optimal solutions of the original problem (or even render the model infeasible). Accordingly, in the computational study we consider the four single SBCs and the three compatible pairwise combinations with SBC1 (SBC1&2, SBC1&3, SBC1&4), resulting in seven SBC settings in total.

3.4 The complexity of UAMSP-R

This section aims to analyze the complexity of UAMSP-R.

Theorem 1 *The studied UAMSP-R is strongly NP-hard.*

Proof 1 Consider a special case where each additive manufacturing machine can only produce one part at a time. In this scenario, UAMSP-R simplifies to an unrelated parallel machine scheduling problem with release times, denoted as $Rm|r_j|C_{\max}$. Since it has been proved that the unrelated parallel machine scheduling problem with release times is strongly NP-hard (Li et al., 2022; Pei et al., 2020), it follows that UAMSP-R, represented as $Rm|batch\{AM\}, rt_r, SET_i|C_{\max}$, is also strongly NP-hard. □

4 Solution method

The UAMSP-R problem is NP-hard, which makes it difficult for a general-purpose MILP solver to efficiently handle large-scale instances. To overcome this challenge, we develop a high-efficiency LBBD algorithm that decomposes the original MILP model into a MP and multiple SPs.

The MP focuses on determining the parts grouping and assignment to machines while accounting for the capacities and characteristics of the AM machines. Given the assignments obtained from the MP, the SPs are then formulated for each machine to determine the optimal sequencing of batches so as to minimize the overall makespan. This separation enables the algorithm to exploit the problem structure and handle large-scale instances more effectively.

The LBBD algorithm proceeds iteratively through information exchange between the MP and the SPs. In each iteration, the MP generates a candidate solution that specifies batch assignments. This solution is then passed to the SPs, which are solved independently to obtain the optimal schedules on each machine. Based on the SP solutions, two types of cuts are generated during the iterative procedure. Feasibility cuts are added when any SP is infeasible, to eliminate from the MP those assignments that would lead to infeasible schedules. Optimality cuts, on the other hand, are generated when all SPs are feasible; they convey the makespan information obtained from the SPs back to the MP, thereby tightening the lower bound and guiding the search toward the global optimum. This mechanism ensures both the feasibility and optimality of the final solution. The process continues until no violated cuts remain and the global optimal solution is reached or a predefined termination condition is satisfied.

4.1 Master problem

According to the process of the LBBD algorithm, the MP of our problem can be formulated as follows:

$$\min C_{max}$$
$$s.t.(2)-(7),(13),(16),(19) \quad and \quad to$$
$$x_{ibr}, v_{ib} \in \{0,1\} \tag{24}$$
$$C_{\max}, PT_{ib} \geq 0 \tag{25}$$
$$CUTS \tag{26}$$

Constraints (26) represent the cuts pool, which is enumerative and can be generated by solving the corresponding SP. The MP determines the part assignment and grouping decision x_{ibr}. Subsequently, a feasible solution can be derived by solving the SPs when the binary variable x_{ibr} is provided by the MP. The SPs correspond to a series of (up to $|I|$) $1|rt_r|C_{\max}$ problems. Given the characteristics of the AM process and considerations for computational efficiency, we propose several feasibility cuts and optimality cuts. Furthermore, we develop an ERD rule algorithm to efficiently address the $1|rt_r|C_{\max}$ problem. Additionally, a heuristic algorithm for the original problem is developed to generate an initial solution, offering an upper bound, which is provided to the LBBD algorithm.

4.2 Subproblems

Based on the batch-to-machine assignments determined in the MP, the SPs are designed to obtain the optimal scheduling sequence of batches on each machine. Let v^*_{ib}, PT^*_{ib}, and $C^*_{\max}$ denote the solutions derived from the MP in iteration κ, where v^*_{ib} represents the assignment of batch b to machine i. Accordingly, $|I|$ independent subproblems are generated, each corresponding to one machine and involving only the non-empty batches assigned to it. For instance, for machine i, only batches satisfying $v^*_{ib} = 1$ are included in its subproblem. Given a fixed assignment v^*_{ib}, the sequencing of batches on one machine does not affect the schedules of other machines. Therefore, all $|I|$ subproblems can be solved independently.

We define a subset J_i of the non-empty batches that are assigned to machine i, i.e., $J_i = \{b \mid v^*_{ib} = 1, \forall b \in B\}$. We define a binary sequencing variable $\eta_{bb'}$ taking value 1 if batch b' is processed immediately after batch b. Then, the SP for machine i (SP_i) can be

formulated as follows:

$$\min \ C_{\max} \tag{1}$$

$$\delta_{bb'} + \delta_{b'b} \leq 1, \quad \forall b, b' \in J_i \tag{27}$$

$$C_{b'} \geq C_b + PT_{b'} - L(1 - \eta_{be}), \quad \forall b, b' \in J_i \tag{28}$$

$$C_b \geq PT_b + \max_{r \in R} x^*_{ibr} rt_r, \quad \forall b \in J_i \tag{29}$$

$$C_{\max} \geq C_b, \quad \forall b \in J_i \tag{30}$$

$$\eta_{be} \in \{0, 1\}, \ C_b \geq 0, \quad \forall b, b' \in J_i \tag{31}$$

Constraints (27) ensure that no batch b is assigned as both the predecessor and successor of another batch at the same time. Constraints (28) and (29) govern the completion-time relationships between consecutive batches, requiring that the start of each batch occur no earlier than the completion of its predecessor and its respective setup period. Constraints (30) define the overall makespan, while Constraints (31) state the feasible ranges of the variables.

We have previously mentioned that SPs correspond to a series of (up to m) $1|rt_r|C_{\max}$ problems. The ERD rule, which efficiently solves the SPs optimally in polynomial time, is detailed in Algorithm 1.

Algorithm 1 ERD rule for SPs

Require: x_{ibr}, PT_{ib}, rt_r
Ensure: $C_{\max}$
Let $r_{ib} \leftarrow \max_{r \in R}\{ rt_r \cdot x_{ibr} \}$ be the release time of batch b on machine i.
Denote t_{ib} as the completion time of batch b on machine i.
Define BS_i as the set of batches on machine i.
for all $i \in I$ **do**
  Sort BS_i in nondecreasing order of r_{ib} to obtain BS'_i.
  $t_{i1} \leftarrow r_{i1} + PT_{i1}$.
  for $b \leftarrow 2$ **to** $|BS'_i|$ **do**
    if $r_{ib} \geq t_{i,b-1}$ **then**
      $t_{i,b} \leftarrow r_{ib} + PT_{ib}$.
    else
      $t_{i,b} \leftarrow t_{i,b-1} + PT_{ib}$.
    end if
  end for
end for
$C_{\max} \leftarrow \max_{i \in I}\left(t_{i,|BS'_i|}\right)$

4.3 Heuristic for initial solution

We propose a fast heuristic to quickly obtain an initial solution used in the LBBD algorithm. The heuristic is presented as Algorithm 2.

Algorithm 2 Heuristic for initial solution

Require: Sets R, I, release times rt_r
Ensure: $C_{\max}$
Sort the parts in R in nondecreasing order of rt_r.
for all $r \in R$ **do** ▷ process parts by release time
  $Q \leftarrow \emptyset$ ▷ candidate assignments
  for all $i \in I$ **do** ▷ try on every machine
    if machine i has batches **then**
      for all batches b on machine i **do**
        Temporarily assign part r to batch b on machine i.
        if the assignment is feasible **then**
          Evaluate the resulting $C_{\max}$ and push $\langle i, b, C_{\max}\rangle$ to Q.
        end if
        Revert the temporary assignment.
      end for
    end if
    Temporarily create a new batch on machine i with part r.
    if the assignment is feasible **then**
      Evaluate the resulting $C_{\max}$ and push $\langle i, \text{new}, C_{\max}\rangle$ to Q.
    end if
    Revert the temporary creation.
  end for
  Select $\langle i^*, b^*, C^*_{\max}\rangle \in Q$ with the minimum $C_{\max}$.
  Permanently assign part r to batch b^* on machine i^*, denotes $x_{i^*b^*r} = 1$.
end for
return the final schedule's $C_{\max}$ and the value of x_{ibr}.

4.4 Benders cuts

4.4.1 Feasibility Cuts

Feasibility cut is generated whenever the variable values from MP result in SP becoming infeasible (Verstichel et al., 2015). Feasibility cut may avoid obtaining previously found solutions in next iteration. In this problem, we designate U as the current best-known upper bound of $C_{\max}$. We iteratively generate cuts whenever the objective value of SP is greater than or equal to U. To obtain a good initial solution as the upper bound U, we develop a heuristic algorithm in Section 4.3. In this section, we propose three feasibility cuts.

We define R_i as the set of parts that achieves the optimal maximum completion time $C_{\max}(R_i)$ on machine i, and B_i as the batch set on machine i. $C_{\max}(R_i)$ can be precisely obtained by the ERD rule, and U is generated by the proposed heuristic algorithm. For each machine i, if $C_{\max}(R_i) \geq U$, we avoid the current solution when solving subsequent iterative MP.

Feasibility cut (32) is:

$$\sum_{b \in B_i} \sum_{r \in R_i} x_{ibr} \leq |R_i| - 1 \quad \forall i \in I, C_{\max}(R_i) \geq U \tag{32}$$

Feasibility cut (32) ensures that at least one part in R_i is excluded from machine i in subsequent iterations. The iterative procedure concludes when no more cuts can be generated or when stop condition is reached.

Furthermore, we obtain a condensed set S_i to form a stronger cut. This is done by the following steps. We add a part to a batch set S_i one by one, according to their index and regardless of the area of the batch being exceeded, calculate the approximate completion

time of the batch each time, which can be calculated as $\tilde{C}_{\max}(S_i) = \min_{r\in S_i} rt_r + SET_i + VT_i \cdot \sum_{r\in S_i} v_r + HT_i \cdot \max_{r\in S_i} h_r$.Once the approximate completion time $\tilde{C}_{\max}(S_i)$ is greater than the optimal solution, we generate the following feasibility cut (33):

$$\sum_{b\in B_i}\sum_{r\in S_i} x_{ibr} \le |S_i| - 1 \quad \forall i \in I, \tilde{C}_{\max}(S_i) \ge U \tag{33}$$

Another way, when the parts grouping and assignment are fixed, the subproblem of each machine is $1|rt_r|C_{\max}$, which can be quickly solved by ERD rule. We initialize the batch set K_i with R_i, that is, $K_i = R_i$. For each parts $r \in R_i$, we gradually delete r from K_i one by one, according to their index and we calculate the makespan $C_{max}(K_i)$. Once the value is greater than U, then, we can generate a new feasibility cut (34):

$$\sum_{b\in B_i}\sum_{r\in K_i} x_{ibr} \le |K_i| - 1 \quad \forall i \in I, C_{\max}(K_i) \ge U \tag{34}$$

4.4.2 Optimality Cuts

Optimality cuts iteratively impose new lower bounds on the makespan $C_{\max}$. When the subproblems are solved, the optimality cut (35) is:

$$C_{\max} \ge C^*_{\max}(R_i)\left(\sum_{b\in B_i}\sum_{r\in R_i} x_{ibr} - |R_i| + 1\right) \quad \forall i \in I \tag{35}$$

Here, R_i represents the part set assigned to machine i. If the part assignment remains unchanged in the next iteration, meaning all decision variables x_{ibr} in R_i generated by the MP retain the value of 1, the right-hand side equals to $C_{\max}(R_i)$, which serves as a lower bound on $C_{\max}$. However, if a part assignment variable x_{ibr} changes from 1 to 0, the efficiency of the cut may diminish. Therefore, we try to find an enhanced optimality cut that can also offer valid information when part assignment changes. The optimality cut (36) is:

$$\begin{aligned} C_{\max} \ge C^*_{\max}(R_i) &- \sum_{b\in B_i}\sum_{r\in R_i}(1 - x_{ibr})(r_t + VT_i \cdot v_r + HT_i \cdot h_r) \\ &- \sum_{b\in B_i}(1 - x_{ib})SET_i \quad \forall i \in I \end{aligned} \tag{36}$$

If the same assignment is given to the SPs, then $x_{ibr} = 1$. Consequently, $\sum_{b\in B_i}\sum_{r\in R_i}(1 - x_{ibr})(rt_r + VT_i \cdot v_r + HT_i \cdot h_r) = 0$ and $\sum_{b\in B_i}(1 - x_{ib})SET_i = 0$ in Inequality (36). In this case, the makespan of a SP $C^*_{\max}(R_i)$ provides a new lower bound on the makespan of MP $C_{\max}$. In contrast, if a different assignment is made to the SPs, at least one of the x_{ibr} equals 0. Therefore, the makespan in the subsequent iteration is bounded by the makespan found in the subproblem minus $\sum_{b\in B_i}\sum_{r\in R_i}(1 - x_{ibr})(rt_r + VT_i \cdot v_r + HT_i \cdot h_r)$. If no parts are assigned to the batch in next iteration, i.e., $x_{ib} = 0$, this means the batch is empty and we must subtract the setup time of this batch.

We enumerate types of cuts used in this paper, which are the cuts mentioned above or combinations of these cuts.

These cuts are as follows:

V1: Optimality Cut (35)

V2: Feasibility Cut (32) and Optimality Cut (35)

V3: Feasibility Cut (33) and Optimality Cut (35)
V4: Feasibility Cut (34) and Optimality Cut (35)
V5: Optimality Cut (36)
V6: Feasibility Cut (32) and Optimality Cut (36)
V7: Feasibility Cut (33) and Optimality Cut (36)
V8: Feasibility Cut (34) and Optimality Cut (36)

4.5 The complete procedure of the LBBD algorithm

We summarize the LBBD implemented by branch-and-cut framework within a single search tree and iteratively adding cuts, the method also referred to as *branch-and-check*. This approach is time-saving and enhances efficiency. The complete procedure is presented as Algorithm 3.

Algorithm 3 LBBD implemented by branch-and-check

Require: Initial upper bound and solution x_{ibr} from Algorithm 2
Ensure: $C_{\max}$
Obtain the initial upper bound U and the value of x_{ibr} from Algorithm 2.
for all $i \in I$ **do**
  Verify that the processing times $\{PT_{ib}\}$ on machine i are in non-decreasing order.
  if not **then**
    Sort $\{PT_{ib}\}$ in non-decreasing order and modify the corresponding x_{ibr} accordingly.
  end if
end for
Initialize the MP with the upper bound U.
$CUTS \leftarrow \emptyset$.
while not optimal **and** stopping criterion not met **do**
  Solve the MP (invoke callback on each new incumbent solution).
  for all $i \in I$ **do**
    Solve the SP on machine i using the ERD rule; obtain C_{ib} and $C_{\max}$.
    Generate a new cut and set $U' \leftarrow C_{\max}$ for the current solution.
    if $U' < U$ **then**
      $U \leftarrow U'$.
    end if
    $CUTS \leftarrow CUTS \cup \{\text{new cut}\}$.
  end for
end while
return $C_{\max}$

5 Computational experiment

To evaluate the performance of the proposed formulations and algorithms, we perform extensive numerical experiments in a series of instances. This section presents numerical results on instances obtained by the proposed LBBD algorithm and GUROBI. All experiments are conducted on a personal computer with Intel Core i7-13700 CPU running at 3.40 GHz, along with 32 GB RAM. The proposed methods are implemented in Python 3.7, with GUROBI 11.0.0 as the optimization solver. The total time limit for each instance is set to 1800 seconds of CPU time. Each instance is executed five times, and the results are averaged to ensure robustness and accuracy.

Table 1 Summary of AM machine specifications

Type	VT_i $\left(\text{hr/cm}^3\right)$	HT_i (hr/cm)	SET_i	MH_i (cm)	MA_i $\left(cm^2\right)$	MP_i
1	0.030864	0.7	10	32	800	1
2	0.030864	0.7	12	40	1200	1
3	0.030864	1	10	40	1000	2
4	0.030864	1	12	40	1225	2
5	0.074074	1.4	20	40	625	3
6	0.074074	1.4	30	40	1600	3

Table 2 Summary of part specifications

Parameter	Value
Height, $h_r\,(cm)$	$Uniform$[5, 40]
Area, $a_r\,(cm^2)$	$Uniform$[50, 1000]
Volume, $v_r\,(cm^3)$	$Uniform$[100, $0.1 \cdot a_r \cdot h_r$]
Precision, p_r	[1, 2, 3]
Release time, rt_r	$Uniform$[0, $\frac{P}{\lvert\bar{R}\rvert}$], $P = \frac{\sum_{i\in I}\sum_{r\in R} PT_r^i}{\lvert I\rvert}$,

5.1 Data generation

We derive the additive machine-related and part-related parameters from Kucukkoc (2019) and Li et al. (2017) which include forming time, powder layering time, setup time, the production area, and the maximum height. Table 1 presents a comprehensive summary of the specifications for six types of distinct additive manufacturing machines. Each machine is evaluated based on various performance metrics crucial for assessing their efficiency and precision in manufacturing tasks. Notably, the precision of each machine is rated on a scale from 1 to 3, with higher numbers signifying greater precision. It is important to mention that the time required for manufacturing may increase as higher precision is sought. The machine data for generating instances is sampled with replacement from Table 1, which provides specifications for various AM machine types. Each machine type is selected independently, allowing the possibility of selecting the same machine type multiple times across different instances.

The part specifications are defined as Table 2. The height, is randomly chosen from a uniform distribution [5,40]. The area varies uniformly between 50 to 1000. The volume is selected from a uniform range from 100 to the product of the area and height. Precision, is defined with discrete values of 1, 2, or 3. Lastly, we modify the instances of Forbes et al. (2024) by providing the release time for each part. In particular, each release time rt_r is uniformly distributed in the ranges $U\left[0, \frac{\rho}{|\bar{R}|}P\right]$, where $P = \frac{\sum_{i\in I}\sum_{r\in R} PT_r^i}{|I|}$, and $PT_r{}^i$ denotes the processing time of a batch with a single part r on the available machine i. The $|\bar{R}|$ is the average number of parts that fit into the build platform according to the areas. Although $|B_i|$ on machine i could be set to $|R|$ without loss of generality, we choose to set it to a slightly smaller value based on empirical observations, aiming to improve computational speed.

Table 3 The performance of symmetry-breaking constraints for small-sized instances

SBCs	ObjVal	Bound	Gap	Time	#Opt/#Fea
SBC1	741.06	722.25	1.79%	722.25	2/5
SBC2	**737.89**	737.89	0	24.81	5/5
SBC3	**737.89**	737.89	0	205.00	5/5
SBC4	738.53	736.57	0.30%	674.15	4/5
SBC1&2	**737.89**	737.89	0	37.22	5/5
SBC1&3	**737.89**	737.89	0	688.36	5/5
SBC1&4	**737.89**	737.78	0.02%	888.21	4/5

The experiments consider varying numbers of machines, including 5, 8, 10, and 20, and varying numbers of parts, ranging from 10, 20, 30, 50, 100, 120 to 150. For each combination of machine and part, we randomly generate 5 instances, for a total of $3 \times 7 \times 5 = 105$ instances. We present the average objective value (ObjVal), average lower bound (Bound), average computational time, average gap, and #Opt/#Fea in results. The gap obtained by each method for each instance, computed as

$$\text{Gap}(\%) = \frac{UB - LB}{UB}$$

UB represents the objective value, while LB denotes the lower bound obtained by each method. Additionally, "#Opt/#Fea" represents the number of optimal solutions and the number of feasible solutions obtained for 5 instances of each size.

5.2 Comparison of symmetry-breaking constraints

To compare the performance of symmetry-breaking constraints, we present the results of small-sized, medium-sized, and large-sized instances, which correspond to the combinations of I and R as (2, 20), (5, 50), and (10, 100), respectively. The first column represents the symmetry-breaking constraints or the combination of them in Table 3 - 5. Different symmetry-breaking constraints or combinations thereof were tested, each labeled as SBC1, SBC2, SBC3, SBC4, SBC1&2, SBC1&3, and SBC1&4. We evaluate their performances using LBBD algorithm with the cut V1.

Table 3 reports the performance of LBBD when different symmetry-breaking constraints are added to the master problem for the small instances with $|I| = 2$ and $|R| = 20$. The results show noticeable differences in runtime and solution quality across the tested SBCs. SBC2 performs best overall. It solves all five instances to proven optimality, reaches a zero gap, and has the smallest average runtime at 24.81 s. SBC3 also solves all instances to optimality with zero gap, but it is much slower with an average runtime of 205.00 s. SBC1 performs the worst. It proves optimality in only 2 out of 5 instances and has the largest gap of 1.79%. SBC4 lies between these cases, proving 4 optimal solutions with a gap of 0.30%. For the combined constraints, SBC1&2 also solves all instances to optimality with zero gap, but its average runtime is 37.22 s, which is longer than using SBC2 alone. This suggests that adding SBC1 does not improve performance for this instance size.

Table 4 reports the results of LBBD with different symmetry-breaking constraints for the medium-sized instances with $|I| = 5$ and $|R| = 50$. All settings are challenging at this scale. Most configurations prove optimality for only 1 out of 5 instances within the time limit, and the remaining runs end with gaps in the range of about 7% to 12%. SBC4 is the only

Table 4 The performance of symmetry-breaking constraints for medium-sized instances

SBCs	ObjVal	Bound	Gap	Time(s)	#Opt/#Fea
SBC1	1016.36	953.23	8.77%	1750.56	1/5
SBC2	1014.28	959.03	7.97%	1442.01	1/5
SBC3	1031.51	954.23	10.49%	1454.46	1/5
SBC4	1028.93	953.61	10.40%	1802.10	0/5
SBC1&2	**1012.64**	962.71	7.12%	1442.46	1/5
SBC1&3	1039.17	953.47	11.56%	1450.80	1/5
SBC1&4	1019.07	954.42	9.05%	1537.81	1/5

Table 5 The performance of symmetry-breaking constraints for large-sized instances

SBCs	ObjVal	Bound	Gap	Time(s)	#Opt/#Fea
SBC1	845.16	789.15	7.10%	1812.19	0/5
SBC2	861.73	772.49	10.57%	1813.94	0/5
SBC3	900.54	766.86	15.33%	1800.84	0/5
SBC4	855.34	781.00	9.18%	1802.46	0/5
SBC1&2	**841.95**	799.46	5.49%	1804.50	0/5
SBC1&3	868.34	782.37	10.61%	1801.60	0/5
SBC1&4	853.77	778.34	9.33%	1801.22	0/5

case that does not prove optimality for any instance. SBC1&2 performs best overall. It gives the lowest objective value (1012.64) and the smallest gap (7.12%), and its average runtime (1442.46 s) is comparable to the fastest settings. Among the single-constraint variants, SBC2 is the strongest. It attains an objective value of 1014.28 with a gap of 7.97%. The weakest performance comes from SBC1&3, which has the largest gap (11.56%) and the highest objective value (1039.17).

Table 5 reports the performance of LBBD with different symmetry-breaking constraints for the large instances with $|I| = 10$ and $|R| = 100$. At this scale, none of the configurations proves optimality within the time limit, and all runs end close to the time cap of about 1800 seconds. The main difference therefore comes from solution quality and the resulting optimality gaps. Among all settings, SBC1&2 performs best. It achieves the smallest objective value (841.95), the tightest bound (799.46), and the lowest gap (5.49%). SBC1 is the next best single constraint with a gap of 7.10%, while SBC3 performs the worst with the largest gap (15.33%) and the highest objective value (900.54). The remaining variants fall between these cases, with gaps around 9%–11%. Overall, SBC1&2 remains the most effective choice for improving solution quality on large instances.

Overall, we adopt SBC1&2 in the subsequent algorithmic and cutting experiments. For the small instances, SBC1&2 performs well and solves all cases to proven optimality, with a runtime that is only slightly higher than using SBC2 alone. As the instance size increases, SBC1&2 becomes clearly preferable. It yields the best solution quality among all tested settings on the medium instances and it also gives the smallest optimality gap on the large instances. This consistent advantage on medium and large cases motivates our choice of SBC1&2 as the default symmetry-breaking setting.

5.3 Comparison of LBBD algorithm and MILP model

We present results obtained by our LBBD algorithm, GUROBI solver and the proposed heuristic on all 105 instances containing various combination of different number of machines and parts. In Table 6, $|I|$ means the amount of AM machines and $|R|$ represents the amount of parts. Our algorithm is named by LBBD, the original model solved by GUROBI named by MILP and HA denotes the heuristic described in Subsection 4.3. In Table 6, LBBD applies cut V1 and symmetry-breaking constraint SBC1&2. The last row of each table represents the average values for all columns, except for the "#Opt/#Fea" column, which shows the total sum.

Table 6 reports the results of LBBD, MILP, and HA on a wide range of instance sizes. For $|R| = 10$, both LBBD and MILP solve all instances to proven optimality within a few seconds. As $|R|$ increases, MILP becomes less reliable and often reaches the time limit. For many settings, especially when $|R| \geq 100$, MILP does not return feasible solutions for all instances, which is indicated by "-" in ObjVal and "0/0" in #Opt/#Fea. In contrast, LBBD consistently returns feasible solutions for all tested configurations. It proves optimality for part of the test set under the same time limit, and for larger instances it typically terminates with moderate optimality gaps.

HA is used to quickly generate an initial feasible solution. It produces a feasible solution in about one second, which is useful for the largest instances when exact methods are difficult to solve within the time limit. The summary in the last row shows that LBBD proves 45 optimal solutions out of 105 feasible runs, with an average gap of 5.96% and an average runtime of 1044.14 seconds. MILP proves 18 optimal solutions out of 53 feasible runs, and its average runtime is 1503.13 seconds. Overall, LBBD provides a good balance between robustness and solution quality, while HA complements the exact methods by supplying feasible solutions for large-scale instances at very low computational cost.

5.4 Comparison of different types of Benders cuts

In this section, we present the performance of LBBD with various types of cuts mentioned before. It includes LBBD with various types of cuts along with their corresponding objective values, bounds, computational times, gap (%), number of calls, number of cuts, and the ratio of optimal to feasible solutions using symmetry-breaking constraint SBC1&2.

Table 7 reports the performance of LBBD under different Benders cut variants for the small instances. All cut types solve all five instances to proven optimality. They reach the same objective value and bound (737.89). The main difference is runtime and the effort spent in the Benders loop. V5 is the fastest on average with 16.94 s, followed closely by V7 (17.40 s) and V8 (17.66 s). V3 is also competitive at 20.10 s, while V1, V2, V4, and V6 are slower at around 25–37 s. In terms of iteration effort, V4 generates the largest number of cuts (8289) and requires many calls (2073), whereas V7 and V5 use fewer calls and produce fewer cuts while still achieving the same optimal solutions.

Table 8 reports the performance of LBBD with different Benders cut variants for the medium-sized instances. At this scale, all variants behave similarly in runtime, since the runs are close to the time limit and the average time stays around 1441–1470 s. Each setting proves optimality for 1 out of 5 instances, and the remaining cases terminate with nonzero gaps. The differences are mainly reflected in solution quality and the amount of Benders cuts added. V6 gives the best objective value (1004.95) and a relatively small gap (6.53%). V3 produces

Table 6 Comparison results obtained by different solving methods.

Instances		LBBD					MILP					HA	
$\|R\|$	$\|I\|$	ObjVal	Bound	Gap	Time	#Opt/#Fea	ObjVal	Bound	Gap	Time	#Opt/#Fea	ObjVal	Time
10	2	**420.84**	420.84	0	0.35	5/5	**420.84**	420.84	0	10.49	5/5	449.36	0
	5	**261.26**	261.26	0	0.12	5/5	261.26	**261.26**	0	2.71	5/5	283.80	0
	8	**223.75**	223.75	0	0.17	5/5	**223.75**	223.75	0	2.56	5/5	226.25	0
20	2	**737.89**	737.89	0	37.22	5/5	–	237.77	–	1800.85	0/4	789.60	0.01
	5	**495.83**	490.69	1.38%	364.89	4/5	**495.83**	306.88	32.48%	1800.43	0/5	562.17	0.01
	8	**307.74**	307.74	0	6.98	5/5	**307.74**	282.52	5.27%	1089.87	2/5	354.50	0
30	2	**1222.15**	1190.75	2.84%	1152.27	2/5	–	241.35	–	1800	0/2	1330.08	0.01
	5	**580.23**	575.06	1.21%	372.73	4/5	–	299.27	–	1800.81	0/4	663.70	0.01
	8	**395.49**	394.54	0.17%	387.92	4/5	**395.49**	287.32	23.78%	1468.29	1/5	468.22	0.02
50	5	**1012.64**	962.71	7.12%	1442.46	1/5	–	281.27	–	1802.59	0/3	1130.73	0.04
	8	**671.03**	624.57	8.70%	1101.87	2/5	724.31	289.67	56.12%	1804.12	0/5	741.14	0.09
	10	**567.79**	552.12	3.05%	1170.79	2/5	581.52	240.11	55.80%	1804.71	0/5	651.86	0.14
100	5	**1563.28**	1446.92	8.42%	1475.36	1/5	–	153.55	–	1811.23	0/0	1649.43	0.32
	8	**1038.85**	916.15	11.81%	1800.18	0/5	–	119.81	–	1812.91	0/0	1258.11	0.21
	10	**841.95**	799.46	5.49%	1804.50	0/5	–	86.19	–	1817.14	0/0	921.74	0.27
120	8	**1519.50**	1389.52	9.88%	1800.87	0/5	–	119.69	–	1811.55	0/0	1689.40	0.37
	10	**1262.68**	1134.31	10.63%	1801.08	0/5	–	61.98	–	1814.00	0/0	1358.68	0.51
	20	**693.47**	608.34	12.77%	1800.47	0/5	–	7.38	–	1828.99	0/0	779.87	0.80
150	8	**1727.87**	1593.54	9.91%	1801.48	0/5	–	74.38	-	1818.91	0/0	1834.72	0.67
	10	**1323.06**	1097.32	16.93%	1804.15	0/5	–	11.45	–	1822.43	0/0	1410.90	1.00
	20	**834.96**	726.65	14.78%	1801.15	0/5	–	6.58	–	1841.24	0/0	894.98	1.02
Average/Total		842.96	783.53	5.96%	1044.14	45/105	–	190.19	–	1503.13	18/53	926.15	0.26

Table 7 The performance of Benders cuts for small-sized instances

Types of cuts	ObjVal	Bound	Time	Gap	#Calls	#Cuts	#Opt/#Fea
V1	**737.89**	737.89	37.22	0	2102	4204	5/5
V2	**737.89**	737.89	37.16	0	1636	5025	5/5
V3	**737.89**	737.89	20.10	0	1712	3423	5/5
V4	**737.89**	737.89	27.60	0	2073	8289	5/5
V5	**737.89**	737.89	16.94	0	1392	2783	5/5
V6	**737.89**	737.89	25.44	0	1589	3565	5/5
V7	**737.89**	737.89	17.40	0	1265	2530	5/5
V8	**737.89**	737.89	17.66	0	1311	5244	5/5

Table 8 The performance of Benders cuts for medium-sized instances

Types of cuts	ObjVal	Bound	Time	Gap	#Calls	#Cuts	#Opt/#Fea
V1	1012.64	959.03	1442.46	7.12%	1104	5522	1/5
V2	1010.29	964.77	1469.81	6.99%	1218	6090	1/5
V3	1006.34	964.92	1443.88	6.27%	1187	5014	1/5
V4	1008.03	958.89	1441.04	7.01%	1131	10457	1/5
V5	1007.29	959.16	1442.03	6.84%	873	4363	1/5
V6	**1004.95**	960.26	1441.51	6.53%	843	4250	1/5
V7	1011.17	958.41	1442.74	7.53%	1056	5279	1/5
V8	1011.17	958.43	1442.57	7.53%	1053	10522	1/5

Table 9 The performance of Benders cuts for large-sized instances

Types of cuts	ObjVal	Bound	Time	Gap	#Calls	#Cuts	#Opt/#Fea
V1	841.95	799.46	1804.50	5.49%	538	5374	0/5
V2	878.82	795.60	1801.80	9.67%	457	5571	0/5
V3	856.53	792.61	1800.84	7.95%	189	1882	0/5
V4	848.47	799.11	1801.57	6.26%	377	3939	0/5
V5	843.62	800.15	1801.77	5.56%	490	4894	0/5
V6	**838.58**	801.47	1800.74	4.84%	630	6654	0/5
V7	839.01	799.12	1801.00	5.17%	348	3480	0/5
V8	848.94	795.03	1801.69	6.86%	215	4212	0/5

the smallest gap (6.27%) and also a low objective value (1006.34). By contrast, V7 and V8 have the largest gaps (7.53%) and higher objective values.

Table 9 reports the performance of LBBD with different Benders cut variants for the large-sized instances. At this scale, none of the variants proves optimality within the time limit, and all runs finish close to 1800 s. The comparison therefore focuses on the solution quality and the resulting optimality gaps. V6 gives the best overall performance. It achieves the lowest objective value (838.58) and the smallest gap (4.84%), together with the tightest bound (801.47). V7 and V1 follow with gaps of 5.17% and 5.49%, respectively. By contrast, V2 performs the worst in this table, with the highest objective value (878.82) and the largest

Table 10 Effect of algorithmic components for small-sized instances

Variants	ObjVal	Bound	Gap	Time(s)	#Opt/#Fea
Full LBBD	**737.89**	737.89	0	16.94	5/5
– (SBC1)	**737.89**	959.03	0	14.19	5/5
– (SBC2)	739.21	954.23	1.94%	1445.33	1/5
– (valid inequalities (19))	774.15	245.89	64.08%	1802.65	0/5
– (initial solution)	764.00	733.89	1.80%	387.29	4/5

Note: "– (X)" denotes the ablation variant obtained by removing component X from the full LBBD configuration.

gap (9.67%). Cuts V3 and V8 require fewer calls, while V6 generates more calls and cuts, which is consistent with its stronger bound improvement and better final gap.

Across different instance sizes, all Benders cut variants lead to stable LBBD performance, but the best choice depends on the scale of the instances. For small-sized instances, every variant solves all cases to proven optimality, and the main differences are runtime; in this setting, V5 is the most efficient. For medium-sized instances, the methods show similar runtimes near the time limit, so solution quality becomes the key factor; V6 delivers the best objective value, while V3 attains the smallest gap. For large-sized instances, none of the variants proves optimality within the time limit, and the comparison is driven by the final gaps. V6 achieves the lowest objective value and the smallest gap overall.

5.5 Effect of algorithmic components

We conduct an ablation study to quantify the contribution of each LBBD component under a controlled setting. Following the results in Section 5.2, we use SBC1&2 as the default symmetry-breaking setting because it remains competitive on small instances and performs best on medium and large instances. For each instance scale, we adopt the best-performing Benders cuts configuration identified in Section 5.4. Specifically, V5 is used for the small-sized instances, while V6 is used for both the medium- and large-sized instances. The baseline (Full LBBD) includes four ingredients: (i) symmetry-breaking constraints SBC1&2 imposed in the master problem, (ii) the selected Benders cuts configuration for the corresponding scale, (iii) the valid inequalities introduced in Section 3.3.1, and (iv) an initial feasible solution generated by HA. Starting from this baseline, we construct a set of remove-one variants by deactivating one ingredient at a time, namely removing SBC1, removing SBC2, removing the valid inequalities, and removing the HA initialization. In addition, when the selected cut configuration for a given scale contains a feasibility cut together with other cuts, we also test a variant in which the feasibility cut is disabled. These controlled comparisons allow us to isolate the marginal impact of each ingredient on solution quality (objective value and final optimality gap) and computational efficiency (runtime), thereby providing direct empirical support for the final configuration.

Table 10 reports the ablation results for the small-sized instances when the full LBBD configuration uses V5 as the cut setting. The full method solves all five instances to proven optimality, with zero gap and an average runtime of 16.94 s. Removing SBC1 has little impact on solution quality. The method still solves all instances to optimality and even shows a slightly shorter average runtime (14.19 s), which indicates that SBC1 does not provide a clear benefit at this scale. This observation is consistent with the results in Section 5.2,

Table 11 Effect of algorithmic components for medium-sized instances

Variants	ObjVal	Bound	Gap	Time(s)	#Opt/#Fea
Full LBBD	**1004.95**	960.26	6.53%	1441.51	1/5
– (SBC1)	1005.14	958.43	6.56%	1440.91	1/5
– (SBC2)	1017.46	948.73	9.05%	1536.52	1/5
– (valid inequalities (19))	1100.27	310.76	68.10%	1801.20	0/5
– (feaibility cut (32))	1007.29	959.16	6.84 %	1442.03	1/5
– (initial solution)	1007.58	958.23	7.01 %	1445.03	1/5

Note: "– (X)" denotes the ablation variant obtained by removing component X from the full LBBD configuration.

Table 12 Effect of algorithmic components for large-sized instances

Variants	ObjVal	Bound	Gap	Time(s)	#Opt/#Fea
Full LBBD	**838.58**	801.47	4.84%	1801.69	0/5
– (SBC1)	844.98	798.45	5.95%	1800.97	0/5
– (SBC2)	848.35	787.50	7.65%	1803.12	0/5
– (valid inequalities (19))	893.98	268.10	71.00%	1800.60	0/5
– (feaibility cut (32))	843.62	800.15	5.56 %	1801.77	0/5
– (initial solution)	-	784.57	-	1802.15	0/4

Note: "– (X)" denotes the ablation variant obtained by removing component X from the full LBBD configuration.

where SBC2 performs better than the combination SBC1&2 on small instances. In contrast, removing SBC2 significantly degrades performance. The objective value increases to 739.21, the average gap rises to 1.94%, and only 1 out of 5 instances is solved to proven optimality within the time limit. The effect is even more pronounced when removing the valid inequalities. In that case, the method fails to prove optimality for any instance and terminates at the time limit with a very large gap (64.08%). Finally, removing the HA initial solution also hurts performance. The objective value increases to 764.00, the gap becomes 1.80%, and the number of proven optimal solutions drops to 4 out of 5, with runtime increasing to 387.29 s.

Table 11 reports the ablation results for the medium-sized instances, where the full LBBD configuration adopts V6 as the cut setting. The full method yields the best objective value (1004.95) with a gap of 6.53%, and it proves optimality for 1 out of 5 instances within the time limit. Removing SBC1 slightly worsens the objective value and gap, which increase to 1005.14 and 6.56%, respectively, while the runtime remains almost unchanged. Removing SBC2 causes a much larger degradation. The objective value rises to 1017.46 and the gap increases to 9.05%, together with a longer runtime. The impact of removing the valid inequalities is even more pronounced. The objective value increases to 1100.27, the gap grows to 68.10%, and no instance is solved to proven optimality. The feasibility cut also improves performance. When it is disabled, the objective value increases from 1004.95 to 1007.29 and the gap rises from 6.53% to 6.84%, with comparable runtime. Finally, removing the HA initial solution leads to a small but consistent loss in solution quality, with the gap increasing to 7.01% and the objective value rising to 1007.58.

Table 12 reports the ablation results for the large-sized instances, where the full LBBD configuration uses V6 as the cut setting. At this scale, none of the variants proves optimality

within the time limit, and all feasible runs finish close to 1800 s. The comparison therefore mainly reflects solution quality and the final optimality gap. The full configuration gives the best overall performance, with the lowest objective value (838.58) and the smallest gap (4.84%). Removing SBC1 worsens the results, increasing the objective value to 844.98 and the gap to 5.95%. Removing SBC2 leads to a larger deterioration, with an objective value of 848.35 and a gap of 7.65%, indicating that SBC2 plays a more important role than SBC1 on large instances. The valid inequalities have the largest impact. Without them, the objective value rises sharply to 893.98 and the gap increases to 71.00%, even though the runtime remains at the time limit. Disabling the feasibility cut also reduces performance, increasing the objective value to 843.62 and the gap to 5.56%, but the effect is smaller than removing SBC1 or SBC2. Finally, without the initial solution, LBBD fails to return complete feasible results.

Across the three ablation tables, a consistent pattern emerges on how each enhancement contributes to LBBD. The valid inequalities are the most influential component at every scale. Once they are removed, performance deteriorates sharply, with much larger gaps, worse objective values, and weaker robustness, and this effect becomes particularly severe on large instances. The role of SBC changes with the instance size. On small instances, removing SBC1 has almost no impact, which is consistent with our earlier observation that SBC2 alone can be slightly more efficient than SBC1&2 at this scale. As the problem size increases, SBC2 becomes increasingly important. Removing SBC2 causes a clear loss in solution quality on both medium and large instances, while removing SBC1 has only a minor effect. The additional enhancements also provide measurable benefits. The feasibility cut improves the final objective and gap on medium and large instances, although its impact is smaller than that of the valid inequalities and SBC2. The HA-based initialization mainly improves robustness. Its removal leads to longer runtimes or reduced feasibility, and this effect is most evident on the largest instances, where the variant without the initial heuristic fails to return complete feasible results. Overall, these results support using SBC1&2 together with the valid inequalities and HA initialization as the default configuration, with feasibility cuts included when applicable.

5.6 Case study

This case study is motivated by an aerospace Maintenance, Repair and Overhaul (MRO) supplier that employs SLM to produce small-batch titanium spare parts on demand. Typical parts include functional brackets, sensor mounts, clamps, access covers, duct segments, and protective shrouds used around the airframe and engine periphery. These parts are typically ordered in low volumes and arrive over time as rolling requests, resulting in heterogeneous release times. Accordingly, we adopt additive manufacturing and focus on rapid, responsive production planning.

We consider two representative industrial SLM machines as production resources: EOS M 290 and EOS M 400-4. Machine capability data (e.g., build volume, maximum build height, and tray dimensions/area) are collected from publicly available information on the EOS website. Specifically, the EOS M 290 is reported to have a build volume of $250 \times 250 \times 325$ mm^3, whereas the EOS M 400-4 has a build volume of $400 \times 400 \times 400$ mm^3. Material and process information for Ti-6Al-4V (Ti64) is obtained from EOS material pages.[1] In particular, available layer-thickness options are used to characterize printing precision and to parameterize the machine precision MP_i, which determines whether a machine can satisfy a

[1] https://www.eos.info/zh/metal-solutions/metal-materials/titanium#eos-titanium-ti64

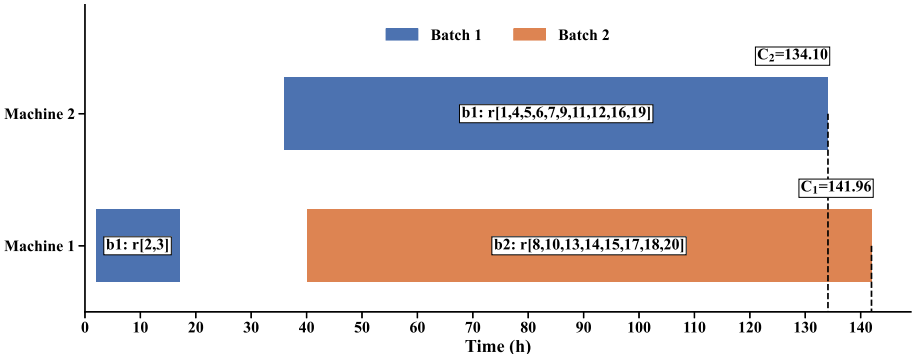

Fig. 3 Gantt chart of the feasible schedule on two machines obtained from HA.

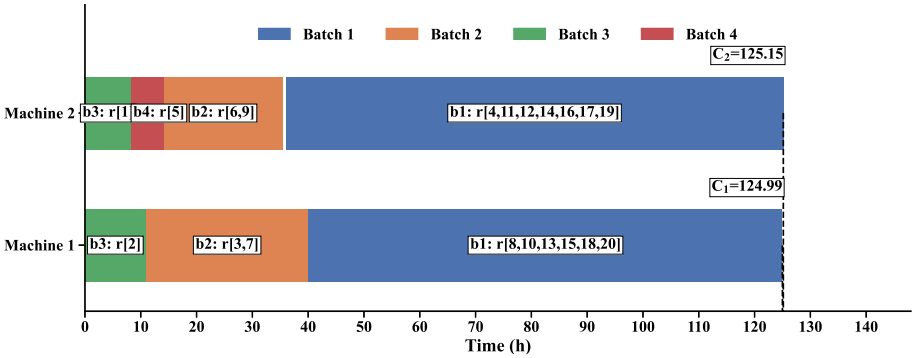

Fig. 4 Gantt chart of the optimal schedule on two machines obtained from LBBD.

part's minimum precision requirement p_r. For clarity, we discretize precision into two levels: parts produced with a layer thickness in 60 μm are classified as precision level 1, whereas parts requiring a layer thickness in 30 μm are classified as precision level 2.

Within the above industrial machine and process boundaries, we construct a demonstrative instance consisting of 20 parts (see Appendix A). The geometric parameters are designed to reflect the heterogeneity commonly observed in aerospace MRO spares, while the release times emulate rolling order arrivals. Parts are grouped into batches on the two machines subject to tray-area and build-height constraints, and precision eligibility rules ensure that a part can only be assigned to machines that satisfy its minimum precision threshold. Therefore, this instance represents an industrially grounded AM scheduling in SLM process for evaluating the proposed approach.

A schedule consists of (i) assigning parts to batches and machines, and (ii) sequencing the batches on each machine. The objective is to minimize the makespan, defined as the maximum completion time among all machines. We first generate a feasible schedule using a heuristic; the resulting makespan is 141.96 h, as shown in Figure 3. Although feasible, this schedule is not guaranteed to be optimal. We then solve the same instance using the proposed LBBD algorithm. LBBD returns a globally optimal schedule with a makespan of 125.15 h, improving the heuristic solution by 16.80 h (approximately 11.8%). Figure 4 illustrates the LBBD optimal schedule. Each colored bar represents the processing of one batch on a machine; the label "$b:r[\cdot]$" indicates that the bar corresponds to batch b and lists

the set of parts assigned to that batch. The dashed vertical lines mark the completion times C_i of the machines. Therefore, the makespan of the instance is $\max_{i \in I} C_i = 125.15$ h.

The results indicate that the decisions of HA may induce longer idle times and/or workload imbalance across machines, which increases the completion time. In contrast, LBBD jointly optimizes batch formation and sequencing, producing a more compact schedule and a reduced makespan. Importantly, explicitly accounting for release times is essential for reliable planning in AM: because an SLM build can only start after all parts assigned to the batch are released, batching rules that focus solely on maximizing bed utilization may inadvertently delay batch starts and increase the makespan. Joint optimization helps balance bed utilization with release-time alignment, thereby improving throughput and schedule robustness in time-sensitive MRO environments.

6 Conclusion

This paper introduces unrelated additive manufacturing machine scheduling problem with different release times with the aim of minimizing makespan. We formulate this problem as a MILP and propose formulation enhancements including valid inequalities, and symmetry-breaking constraints. We derive the NP-hardness of UAMSP-R. Due to the complexity of UAMSP-R, a LBBD algorithm with callback is developed. The proposed LBBD decomposes the original problem into a master problem and subproblems. The master problem, which simultaneously determines part assignment and grouping with cuts generated by solving a subproblem to sequence batches on each single machine. An ERD rule is designed to efficiently solve the SP. Three feasibility cuts and two optimality cuts are generated and we combine them into eight different types of cuts. A heuristic is developed to obtain a good initial solution for the LBBD algorithm.

Computational experiments are conducted on a set of randomly generated instances covering a wide range of problem sizes. We first evaluate different symmetry-breaking constraints within LBBD. The results show that SBC2 performs best on small instances, while the combination SBC1&2 provides the most reliable and best overall performance on medium- and large-sized instances. Based on this observation, SBC1&2 is adopted as the default setting in the subsequent experiments. We then compare the proposed LBBD algorithm with the original MILP formulation solved by GUROBI on the full test set. The results demonstrate that LBBD is substantially more robust and scalable. It consistently returns feasible solutions and achieves better solution quality within the same time limit, whereas the MILP model frequently reaches the time limit and fails to provide feasible solutions for large instances. Next, we examine the impact of different Benders cut variants. For small instances, all cut types reach proven optimality, and V5 yields the best computational efficiency. For medium and large instances, the comparison is mainly driven by final gaps and objective values under the time limit, and V6 provides the best overall performance. Finally, we conduct a component-wise ablation study on each instance scale. Using SBC1&2 and the best-performing cut configuration for that scale as the baseline, we remove one component at a time, including SBC1, SBC2, the valid inequalities, and the HA initialization. When applicable, we also test disabling feasibility cuts. These controlled experiments quantify the contribution of each ingredient to both solution quality and computational efficiency.

The UAMSP-R problem is an important combinatorial optimization problem with broad practical applications. However, the literature on this problem is limited. In future studies, the problem may be extended to consider other AM scenarios such as part placement, stochas-

Table 13 Machine-related parameters for the two SLM machines.

Machine i	MH_i (cm)	MA_i (cm^2)	MP_i (level)	SET_i (h)	VT_i (h/cm^3)	HT_i (h/cm)
1	32.5	625	2	2	0.074074	1.1
2	40.0	1600	1	2	0.030864	0.6

Table 14 Data for the 20 parts aerospace MRO Ti64 SLM instance.

Part r	v_r (cm^3)	h_r (cm)	a_r (cm^2)	p_r (level)	rt_r (h)
1	42	8.4	38	1	0
2	28	6.2	22	2	0
3	36	7.5	30	2	2
4	55	9.8	44	1	4
5	60	3.2	120	1	6
6	88	11.6	62	1	8
7	120	14.2	70	1	10
8	22	5.5	18	2	12
9	210	16.8	95	1	14
10	110	12.5	58	2	16
11	420	28.0	160	1	18
12	260	30.5	75	1	20
13	65	10.2	40	2	22
14	50	7.0	32	1	24
15	240	18.6	105	1	26
16	520	35.0	190	1	28
17	180	21.0	88	1	30
18	26	6.8	20	2	32
19	600	38.0	210	1	36
20	300	24.0	140	1	40

tic release time. Efficient heuristics are expected to be developed to provide good feasible solutions in acceptable computation times. The LBBD algorithm can be further improved by introducing multiple stronger cuts. As a promising direction for future research, instance space analysis could be incorporated to systematically position our exact algorithm relative to heuristic approaches. Such techniques would enable a deeper understanding of the problem characteristics under which exact or heuristic methods are preferable, thereby complementing the contributions of the present study.

Appendix A Case study data

This appendix reports the input data used in the aerospace MRO SLM case study to facilitate reproducibility. Table 13 summarizes machine-related parameters for the two SLM machines, including the maximum build height MH_i, available tray area MA_i, machine precision level MP_i, setup time SET_i, and the volume- and height-dependent time coefficients (VT_i and HT_i). Table 14 provides the attributes of the 20 parts, where each part r is characterized by its

volume v_r, height h_r, footprint area a_r, minimum precision requirement p_r, and release time rt_r. These parameters are used to enforce build-height and tray-area feasibility, machine–part precision eligibility, and release-time constraints in batch formation and scheduling, and to compute batch processing times following the model definitions in Section 3.

Funding Open access funding provided by The Hong Kong Polytechnic University This work was supported by the National Natural Science Foundation of China (Grant No.72571139, No.52305557), the Qing Lan Project, and Guangdong Basic and Applied Basic Research Foundation (No.2024A1515011930).

Data Availability To enhance transparency and reproducibility, all benchmark instances and results in this paper have been made publicly available at: https://github.com/OurGroupNUAA/UAMSP-R. For each instance, we report the optimal objective value when available; for instances where optimality was not reached within the time limit, the best-known objective value is provided.

Declarations

Conflicts of Interest No potential conflict of interest was reported by the authors.

References

Alicastro, M., Ferone, D., Festa, P., Fugaro, S., & Pastore, T. (2021). A reinforcement learning iterated local search for makespan minimization in additive manufacturing machine scheduling problems. *Computers & Operations Research, 131*, Article 105272.

Altekin, F. T., & Bukchin, Y. (2022). A multi-objective optimization approach for exploring the cost and makespan trade-off in additive manufacturing. *European Journal of Operational Research, 301*(1), 235–253.

Bahnini, I., Rivette, M., Rechia, A., Siadat, A., & Elmesbahi, A. (2018). Additive manufacturing technology: the status, applications, and prospects. *The International Journal of Advanced Manufacturing Technology, 97*, 147–161.

Benders, J. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik, 4*, 238–252.

Bruni, M. E., Khodaparasti, S., & Moshref-Javadi, M. (2022). A logic-based Benders decomposition method for the multi-trip traveling repairman problem with drones. *Computers & Operations Research, 145*, Article 105845.

Chen, J., Ma, W., Ye, X., & Zhao, Z. (2023). Logic-based Benders decomposition for order acceptance and scheduling in distributed manufacturing. *Advanced Engineering Informatics, 58*, Article 102228.

Chen, J., Ye, X., Ma, W., & Xu, D. (2024). Logic-based Benders decomposition for order acceptance and scheduling on heterogeneous factories with carbon caps. *Computers & Operations Research, 168*, Article 106706.

Fanjul-Peyro, L., Ruiz, R., & Perea, F. (2019). Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. *Computers & Operations Research, 101*, 173–182.

Fera, M., Fruggiero, F., Lambiase, A., Macchiaroli, R., & Todisco, V. (2018). A modified genetic algorithm for time and cost optimization of an additive manufacturing single-machine scheduling. *International Journal of Industrial Engineering Computations, 9*(4), 423–438.

Fera, M., Macchiaroli, R., Fruggiero, F., & Lambiase, A. (2020). A modified tabu search algorithm for the single-machine scheduling problem using additive manufacturing technology. *International Journal of Industrial Engineering Computations, 11*(3), 401–414.

Forbes, M., Harris, M., Jansen, H., van der Schoot, F., & Taimre, T. (2024). Combining optimisation and simulation using logic-based Benders decomposition. *European Journal of Operational Research, 312*(3), 840–854.

Fowler, J. W., & Mönch, L. (2022). A survey of scheduling with parallel batch (p-batch) processing. *European Journal of Operational Research, 298*(1), 1–24.

Gardan, J. (2016). Additive manufacturing technologies: state of the art and trends. *International Journal of Production Research, 54*(10), 3118–3132.

Gedik, R., Rainwater, C., Nachtmann, H., & Pohl, E. A. (2016). Analysis of a parallel machine scheduling problem with sequence dependent setup times and job availability intervals. *European Journal of Operational Research, 251*(2), 640–650.

Guo, C., Bodur, M., Aleman, D. M., & Urbach, D. R. (2021). Logic-based benders decomposition and binary decision diagram based approaches for stochastic distributed operating room scheduling. *INFORMS Journal on Computing, 33*(4), 1551–1569.

Guo, P., & Zhu, J. (2023). Capacity reservation for humanitarian relief: A logic-based Benders decomposition method with subgradient cut. *European Journal of Operational Research, 311*(3), 942–970.

Hedenstierna, C. P. T., Disney, S. M., Eyers, D. R., Holmström, J., Syntetos, A. A., & Wang, X. (2019). Economies of collaboration in build-to-model operations. *Journal of Operations Management, 65*(8), 753–773.

Hooker, J. N. (2007). Planning and scheduling by logic-based Benders decomposition. *Operations research, 55*(3), 588–602.

Hooker, J. N., & Ottosson, G. (2003). Logic-based Benders decomposition. *Mathematical Programming, 96*(1), 33–60.

Hu, K., Che, Y., & Zhang, Z. (2022). Scheduling unrelated additive manufacturing machines with practical constraints. *Computers & Operations Research, 144*, Article 105847.

Kanishka, K., & Acherjee, B. (2023). Revolutionizing manufacturing: A comprehensive overview of additive manufacturing processes, materials, developments, and challenges. *Journal of Manufacturing Processes, 107*, 574–619.

Kapadia, M. S., Uzsoy, R., Starly, B., & Warsing, D. P., Jr. (2022). A genetic algorithm for order acceptance and scheduling in additive manufacturing. *International Journal of Production Research, 60*(21), 6373–6390.

Khorram Niaki, M., & Nonino, F. (2017). Additive manufacturing management: a review and future research agenda. *International Journal of Production Research, 55*(5), 1419–1439.

Kucukkoc, I. (2019). Milp models to minimise makespan in additive manufacturing machine scheduling problems. *Computers & Operations Research, 105*, 58–67.

Li, Q., Kucukkoc, I., & Zhang, D. Z. (2017). Production planning in additive manufacturing and 3d printing. *Computers & Operations Research, 83*, 157–172.

Li, Q., Zhang, D., Wang, S., & Kucukkoc, I. (2019). A dynamic order acceptance and scheduling approach for additive manufacturing on-demand production. *The International Journal of Advanced Manufacturing Technology, 105*, 3711–3729.

Li, Y., Côté, J. F., Coelho, L. C., & Wu, P. (2022). Novel efficient formulation and matheuristic for large-sized unrelated parallel machine scheduling with release dates. *International Journal of Production Research, 60*(20), 6104–6123.

Mao, Z., Fu, E., Huang, D., Fang, K., & Chen, L. (2024). Combinatorial benders decomposition for single machine scheduling in additive manufacturing with two-dimensional packing constraints. *European Journal of Operational Research, 317*(3), 890–905.

Naderi, B., & Roshanaei, V. (2020). Branch-relax-and-check: A tractable decomposition method for order acceptance and identical parallel machine scheduling. *European Journal of Operational Research, 286*(3), 811–827.

Naderi, B., & Roshanaei, V. (2022). Critical-path-search logic-based benders decomposition approaches for flexible job shop scheduling. *INFORMS Journal on Optimization, 4*(1), 1–28.

Nascimento, P. J., Silva, C., Antunes, C. H., & Moniz, S. (2024). Optimal decomposition approach for solving large nesting and scheduling problems of additive manufacturing systems. *European Journal of Operational Research, 317*(1), 92–110.

Oh, Y., Witherell, P., Lu, Y., & Sprock, T. (2020). Nesting and scheduling problems for additive manufacturing: A taxonomy and review. *Additive Manufacturing, 36*, Article 101492.

Pei, Z., Wan, M., & Wang, Z. (2020). A new approximation algorithm for unrelated parallel machine scheduling with release dates. *Annals of Operations Research, 285*(1), 397–425.

Rahmaniani, R., Crainic, T. G., Gendreau, M., & Rei, W. (2017). The Benders decomposition algorithm: A literature review. *European Journal of Operational Research, 259*(3), 801–817.

Ransikarbum, K., Ha, S., Ma, J., & Kim, N. (2017). Multi-objective optimization analysis for part-to-printer assignment in a network of 3d fused deposition modeling. *Journal of Manufacturing Systems, 43*, 35–46.

Roshanaei, V., Luong, C., Aleman, D. M., & Urbach, D. (2017). Propagating logic-based Benders' decomposition approaches for distributed operating room scheduling. *European Journal of Operational Research, 257*(2), 439–455.

Thompson, M. K., Moroni, G., Vaneker, T., Fadel, G., Campbell, R. I., Gibson, I., Bernard, A., Schulz, J., Graf, P., Ahuja, B., et al. (2016). Design for additive manufacturing: Trends, opportunities, considerations, and constraints. *CIRP annals, 65*(2), 737–760.

Tofail, S. A., Koumoulos, E. P., Bandyopadhyay, A., Bose, S., O'Donoghue, L., & Charitidis, C. (2018). Additive manufacturing: scientific and technological challenges, market uptake and opportunities. *Materials today, 21*(1), 22–37.

Verstichel, J., Kinable, J., De Causmaecker, P., & Berghe, G. V. (2015). A combinatorial Bender' decomposition for the lock scheduling problem. *Computers & Operations Research, 54*, 117–128.

Wang, S., Wu, R., Chu, F., & Yu, J. (2022). Unrelated parallel machine scheduling problem with special controllable processing times and setups. *Computers & Operations Research, 148*, Article 105990.

Wu, P., Wang, Y., & Chu, C. (2024). Logic-based Benders decomposition for bi-objective parallel machine selection and job scheduling with release dates and resource consumption. *Computers & Operations Research, 164*, Article 106528.

Xiong, F., Jing, L., Xiong, N., & Xiang, C. (2025). Enhanced logic-based Benders decomposition methods for the distributed heterogeneous non-permutation flow shop scheduling problem. *International Journal of Production Research*: 1–38 .

Xiong, F. & Liu, H. (2025). Logic-based benders decomposition methods for the distributed flexible job shop scheduling problem. *European Journal of Operational Research* .

Yılmaz, Ö. F. (2020). Examining additive manufacturing in supply chain context through an optimization model. *Computers & Industrial Engineering, 142*, Article 106335.

Ying, K. C., Fruggiero, F., Pourhejazy, P., & Lee, B. Y. (2022). Adjusted iterated greedy for the optimization of additive manufacturing scheduling problems. *Expert Systems with Applications, 198*, Article 116908.

Zhang, Z., Song, X., Huang, H., Zhou, X., & Yin, Y. (2022). Logic-based Benders decomposition method for the seru scheduling problem with sequence-dependent setup time and DeJong's learning effect. *European Journal of Operational Research, 297*(3), 866–877.

Zipfel, B., Neufeld, J., & Buscher, U. (2024). An iterated local search for customer order scheduling in additive manufacturing. *International Journal of Production Research, 62*(3), 605–625.