CIRP Journal of Manufacturing Science and Technology 57 (2025) 100–115

Contents lists available at ScienceDirect

CIRP Journal of Manufacturing Science and Technology

journal homepage: www.elsevier.com/locate/cirpj

Production scheduling of additively manufactured metal parts☆

Kuo-Ching Ying [a], Shih-Wei Lin [b,c,d,1], Pourya Pourhejazy [e,*], Fei-Huan Lee [a,f]

[a] Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei 10608, Taiwan
[b] Department of Information Management, Chang Gung University, Taoyuan 333, Taiwan
[c] Department of Emergency Medicine, Keelung Chang Gung Memorial Hospital, Keelung 204, Taiwan
[d] Department of Industrial Engineering and Management, Ming Chi University of Technology, New Taipei City 243, Taiwan
[e] Department of Industrial Engineering, UiT, The Arctic University of Norway, Lodve Langesgate 2, Narvik 8514, Norway
[f] Everlight Electronics Co., Ltd., IT Division, No. 6-8, Zhonghua Road, Shulin District, New Taipei City 238, Taiwan

ARTICLE INFO

Keywords:
Additive manufacturing
3D printing
Production planning
Laser Powder Bed Fusion (PBF-LB/M)
Optimization
Sustainable Development Goals: SDG 9

ABSTRACT

The production of metal products is one of the main areas where supply chains benefit from adopting additive manufacturing (AM). Optimizing the production process facilitates the widespread adoption of AM by improving know-how and reducing costs. This study offers a twofold contribution to facilitate the implementation of Additive Manufacturing Scheduling Problems (AMSPs) for producing metal parts. First, two mathematical formulations are proposed to enable the use of commercial solvers to optimize small- and medium-sized AMSPs. Second, a highly competitive solution algorithm called Tweaked Iterative Beam Search (TIBS) is developed to find (near-) optimal solutions to industry-scale problems. A total of 225 instances of various workloads are considered for numerical experiments, and the algorithm's performance is evaluated, comparing it with the baselines. In 165 small and medium-sized instances, TIBS yielded 71 optimal solutions and 106 best-found solutions. For large-scale cases, all of the best-found solutions were obtained by TIBS. The statistical results support the significance of the outcomes in the optimization performance.

1. Introduction

Traditional manufacturing based on subtractive methods such as cutting, turning, milling, grinding, and drilling, as well as forming methods like forging, casting, welding, and metallurgy, are limited in that the production process becomes more expensive as (1) the material and design complexities of products/parts increase and (2) the economy-of-scale decreases. As a disruptive new technology, Additive Manufacturing (AM), also known as 3D printing, offers much-needed support in addressing complexity- and economy-of-scale-related issues and brings about supply chain benefits [1]. AM consists of layer-by-layer addition of compound material based on Computer-Aided Design (CAD) of 3D models [2,3]. The major advantages of AM in production management include design and material flexibility, resource and material efficiency (resulting in less waste), high precision, and a short development cycle from prototyping to production [4]. AM has already gained widespread adoption for the mass production of complex parts as well as brackets in various industries.

In metal AM, processes are systematically categorized based on the energy source employed, namely laser, electron beam, or electric arc, and the physical form of the metallic material utilized, either in powdered or wire form [5]. Large enterprises from the automotive and aviation industries have adopted AM technologies —notably Laser Powder Bed Fusion (PBF-LB/M) and Directed Energy Deposition (DED) that hold over 93 % of the market share [6]— to produce metal parts with good quality and fast response time, respectively. The major difference between the two most common metal AM approaches is that PBF-LB/M is superior to DED in developing intricate internal structures while DED supports greater material- and build-size efficiency [7]. Recognizing the importance of time, cost, and know-how in adopting a

☆ The corresponding author, Pourya Pourhejazy would like to acknowledge the financial support from the Interreg Aurora Program for implementing DED AM in future manufacturing – IDiD project with grant reference number 20358021. The work of Shih-Wei Lin was partially supported by the National Science and Technology Council, the Republic of China (Taiwan) under Grant MOST111-2410-H-011-020-MY3.

* Corresponding author.
E-mail addresses: kcying@ntut.edu.tw (K.-C. Ying), swlin@mail.cgu.edu.tw (S.-W. Lin), pourya.pourhejazy@uit.no (P. Pourhejazy), tonylee0912@gmail.com (F.-H. Lee).
[1] Co-first author.

https://doi.org/10.1016/j.cirpj.2025.01.005

new technology, further developments in production operations management are required to accelerate the adoption of metal AM by Small and Medium Enterprises (SMEs). While metal AM has seen some technical developments in the academic literature, the production planning and control of metal 3D printing operations have only recently received attention [8].

Additive Manufacturing-based Production Scheduling (AMPS) differs from subtractive-based production scheduling in that the classic variant focuses on determining the job sequence that optimizes production performance. AMPS problems involve determining the part groups, their placement on the build plate, and the sequence of jobs in every batch; this makes AMPS problems more complex than their classic counterparts. Finding efficient ways to optimize AMPSs is a prerequisite for broader industrial adoption.

The existing literature on AMPSs can be investigated in 12 categories considering the number of parts (Single *vs.* Multiple), the number of base plates (Single *vs.* Multiple), and the number of machines (Single-machine, Identical Multiple machines, and Non-identical Multiple machines) as the differentiating attributes. The present research is focused on the M/M/S variant. For a comprehensive review of these models and their variants, we refer interested readers to recent reviews by [9–11].

From the most relevant literature, [12,13] improved Genetic and Tabu Search algorithms for optimizing single-machine AMPSs. [4] developed a generic mixed-integer linear programming formulation to minimize the maximum completion time in AMPSs with both single and parallel machines. [14] proposed a mixed-integer programming formulation and a heuristic algorithm to solve AMPSs. These studies considered multiple products but with single base plates. Only three studies considered the M/M/S variant, which is suitable for metal production using both DED and PBF-LB/M technologies. [15] developed an improved version of the reinforcement learning iterative local search-based metaheuristic by enhancing the neighborhood search module using the Q-learning variable. [16] studied the cloud-based AMPS, which matches manufacturing resources considering multi-task requirements to improve the utilization of idle resources, and reduce costs. In their study, a multi-3D printing task scheduling was modeled in the form of mixed integer linear programming to find the minimum average cost of materials per unit volume. More recently, [3] developed an improved version of the iterated greedy algorithm to solve AMPS in the M/M/S setting. They did not provide a mathematical formulation of the problem.

In the most recent study, [17] developed a branch-and-price algorithm enhanced with a column-generation technique to optimize the single stereolithography machine AMPS. In addition to the differences in materials used (thermoplastic-like materials *vs.* metals) and the manufacturing techniques employed, the processing time of a batch in stereolithography and PBF-LB/M differs. This difference is contingent on the individual geometry and various process parameters, including layer thickness, laser power, and scanning velocity. Recognizing this distinction is crucial for both modeling the mathematical model and developing the solution algorithm. Tailored optimization approaches can significantly broaden the industrial applications of this group of scheduling problems.

The main contribution of this study is to develop a novel optimization algorithm based on Beam Search to effectively and efficiently solve the M/M/S variant of AMPS problems. This represents a new application area for Beam Search as a relatively underexplored optimization algorithm. Additionally, a new mathematical formulation is proposed for solving small and medium-sized instances using commercial (exact) solvers. To achieve the study objectives, a literature review is first conducted in Section 2 to review the latest developments in the field. The mathematical formulations and solution algorithm are then presented in Sections 3–4, respectively. Next, numerical experiments are reported in Section 5, followed by drawing general conclusions in Section 6.

2. Literature review

There are two major streams in the AMPS literature. The first group of studies is concerned with extending the problem definition and/or formulation, while the second group develops more efficient means of solving the problem. [18] is one of the first studies to extend the AMPS problem, including additional variables, notably orientation selection and two-dimensional packing. [19] developed a genetic algorithm to solve the integrated problems of order acceptance and AMPS. [20] developed a heuristic method for solving AMPS, considering technological constraints. [15] developed a reinforcement learning-based iterated local search to minimize the maximum completion time in the basic AMPS problems. [21] developed a non-dominated sorting genetic algorithm for a multi-objective variant of AMPS with parallel PBF-LB/M machines. [22] developed an exact method for multi-objective optimization of small instances to schedule the same AM process category. [14] developed a mathematical formulation and a simple heuristic method based on a local search for single-machine batch scheduling and assembly in a generic AMPS setting. Focusing on a practical contribution, [23] integrated the time-of-use system into AMPS to account for time-varying electricity prices.

[24] developed a general framework for AMPS. [25] introduced the AMPS problem integrated with distribution variables and solved the problem using an improved version of the branch-and-price algorithm. [26] introduced a collaborative batching problem in distributed AMPS. [3] modified the iterated greedy algorithm for the optimization of single-machine AMPS problems. [27,28] developed exact optimization approaches for identical parallel machine AMPS, considering multiple processing alternatives. [29] developed a multi-agent-based AMPS based on a genetic algorithm for medical 3D printing. [16] developed a generic heuristic strategy for the scheduling of 3D printing tasks on a cloud platform. [30] introduced an integrated optimization approach to schedule build and post-processes for decomposed parts. [31] developed an adaptive large neighborhood search algorithm for solving AMPS with unrelated parallel machines. [32] developed a mathematical model and used a commercial solver to solve the single-machine AMPS with sequence-dependent setup times and multi-material parts, considering small instances.

Most recently, [33] developed an iterated local search for the scheduling of customer orders in the AM context. [34] compared the centralized and distributed scheduling of robotic-based AM using an improved genetic algorithm and simulations, respectively. [35] developed an integrated parts nesting and production scheduling considering AMPS with the order due dates. [36] developed the iterated epsilon-greedy algorithm for the integrated optimization of part-packing and build-scheduling problems, considering parallel machines. [37] developed a reinforcement learning iterated greedy algorithm for integrated scheduling of two-stage AM and assembly operations. Reference [17] developed a branch-and-price algorithm to solve AMPS for stereolithography machines that use thermoplastic-like materials as feedstock.

Table 1 summarizes the 25 articles discussed above, highlighting each study's methodological approach, the number of objective functions examined, and additional features such as whether their model addressed other value chain links and the demand type. As shown in Table 1, most of the existing literature used quantitative approaches to solve AMPSs with single-objective and static demand. Besides, these studies exclusively focused on parts production scheduling, disregarding coordinated activities within the value chain.

3. Problem statement and mathematical formulations

3.1. Problem statement

Since PBF-LB/M machines used for producing metal parts are expensive, especially for SMEs to acquire, the production process is

Table 1
Summary of the characteristics of AMPS literature.

Author	Quantitative approach	Single-objective	Value chain requirements	Static Demand
Ying et al.[3]	✓	✓		✓
Arık[13]	✓	✓	✓	✓
Alicastro et al.[14]	✓	✓		✓
Wu et al.[15]	✓	✓		
Liu et al.[16]	✓	✓		✓
Che et al.[17]	✓	✓		✓
Kapadia et al.[18]	✓	✓		✓
Aloui and Hadj-Hamou[19]	✓	✓		✓
Rohaninejad et al.[20]	✓			✓
Altekin and Bukchin[21]	✓			✓
Karimi et al.[22]	✓			✓
De Antón et al.[23]				✓
He et al.[24]	✓		✓	✓
Zehetner et el.[25]	✓	✓	✓	✓
Kim and Kim[26]	✓	✓		✓
Kim and Kim[27]	✓	✓		✓
He et al.[28]	✓			✓
Oh and Cho[29]	✓	✓		✓
Hu et al.[30]	✓	✓		✓
Toksarı and Toğa[31]	✓			✓
Zipfel et al.[32]	✓	✓		✓
Poudel et al.[33]	✓	✓		✓
Nascimento et al.[34]	✓	✓		✓
Lee and Kim[35]	✓	✓		✓
Ying and Lin[36]	✓	✓	✓	✓

often outsourced. Orders received from decentralized customers must be regrouped and assigned to machines in the form of different job batches. In this process, it is important to ensure that the machine's capacity, specifically its height, and area, are taken into consideration. The production rates of AM processes are slower than those of conventional manufacturing techniques such as forging or casting. The production rate depends on the batch size and the geometrical complexity. The 3D print processing time of a batch depends on the sequence-independent setup operations required before processing a new round of production, the required time for adding material per unit volume, the time used for powder-layering each layer, the total volume of parts assigned to it, and the maximum height of parts assigned to it; how to combine and distribute the parts into job batches to minimize the maximum completion time (makespan) of the orders highlights the need for AMPS. Overall, the AMPS process involves sorting parts into specific job batches to be processed on a 3D printing machine, as shown in Fig. 1.

In the illustrative example, a single 3D printing machine based on PBF-LB/M uses a high-power laser beam to melt the metal powder and form the desired shape and structure. The problem under study, denoted by $1|batch\{AM\}|C_{\max}$, is a single-machine AMPS. This study extends the mathematical formulation proposed by [4] and provides a compact formulation, which facilitates the use of commercial solvers (*i.e.*, exact optimizers) for solving small- and medium-sized instances. The following assumptions and mathematical notation are considered in the formulations.

(1) The number of parts and their sizes are deterministic and known. The height of all parts is less than or equal to the maximum height that can be printed by the 3D printer. All parts in the same job batch have the same processing and completion times.
(2) Once a job batch has started to be processed by a 3D printer, parts cannot be reassigned.
(3) The release time of all parts is zero, that is, all parts can be processed immediately at the beginning of the production planning period.
(4) The area of the parts in the tray is determined by the rectangular shape of their projected area, and the parts are designed with a

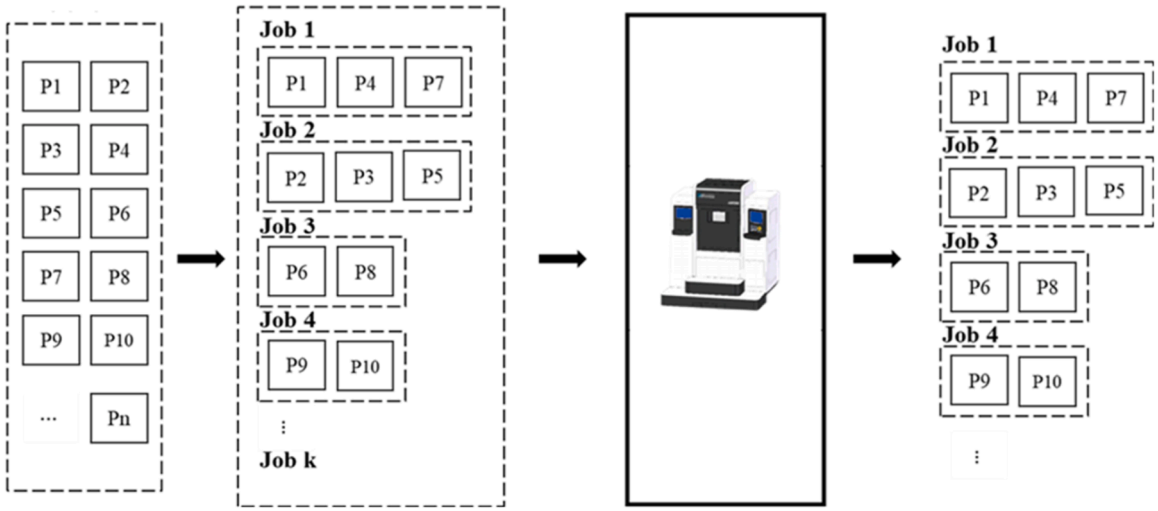

Fig. 1. Visual illustration of additive manufacturing production scheduling.

predetermined placement direction. As a result, the shape and placement direction of the parts are not taken into consideration.
(5) A part can only be assigned to one batch processing and a certain base plate, and the 3D printing machine can only process one batch at a time.
(6) Once started, the batch processing operation cannot be interrupted, which means that the possibility of machine failure or maintenance is ignored.
(7) The defect rate of the parts is negligible, and there is no scrap or reworks.

3.2. Problem formulations

In this subsection, an existing Mixed-Integer Linear Programming (MILP) formulation for the $1|batch\{AM\}|C_{\max}$ problem is first refined and corrected (the F_1 formulation). A novel MILP formulation is then proposed, where symmetry-breaking constraints are introduced to improve the computational efficiency in solving the problem (the F_2 formulation); this will be tested in the numerical experiments. These formulations are referred to as $MILP_{F1}$ and $MILP_{F2}$, respectively. The following notations are used to develop these formulations.

Indices.
iPart, $i \in I$ and $i = 1, 2, \ldots, i_n$.
j Job batch, $j \in J$ and $j = 1, 2, \ldots, j_n$.
Parameters and decision variables.
i_n Number of parts to be processed.
j_n Maximum number of job batches to be processed.
h_i Height of part i.
a_i Area of part i.
v_i Volume of part i.
a_m Maximum area on the build plate; $\max_{i\in I}\{a_i\} \le a_m$.
s_t Setup time required before processing a new round of production.
u_t Time required for molding the material per unit of volume.
h_t Time required for powder build-up per layer.
ψ A sufficiently large constant.
P_j Processing time of job batch j.
C_j Completion time of job batch j.
X_{ji} Binary variable; $= 1$ when part i is assigned to batch j, and $= 0$, otherwise.
Z_j Binary variable; $= 1$ when batch j is active by assigning any parts, and $= 0$, otherwise.
H_j Continuous variable; the maximal height of parts in job batch j.

Considering the relationships between the completion times of job batches, Kucukkoc [4] formulated the $1|batch\{AM\}|C_{\max}$ problem as the MILP formulation shown below.

$$\text{Minimize} \quad \max_{j\in J}\{C_j\} \tag{1}$$

Subject to

$$\sum_{j\in J} X_{ji} = 1; \forall i \in I \tag{2}$$

$$\sum_{i\in I} a_i \cdot X_{ji} \le a_m; \forall j \in J \tag{3}$$

$$\sum_{i\in I} X_{(j+1)i} \le \psi \cdot \sum_{i\in I} X_{ji}; \forall j \in J\backslash\{j_n\} \tag{4}$$

$$C_{(j-1)} + P_j \le C_j; \forall j \in J \tag{5}$$

$$C_0 = 0 \tag{6}$$

$$P_j = s_t \cdot Z_j + u_t \cdot \sum_{i\in I} v_i \cdot X_{ji} + h_t \cdot \max_{i\in I}\{h_i \cdot X_{ji}\}; \quad \forall j \in J \tag{7}$$

$$X_{ji}, \quad Z_j \in \{0,1\}; \forall j \in J; \forall i \in I \tag{8}$$

3.2.1. $MILP_{F1}$

To ensure the proper execution of Kucukkoc's MILP formulation (see [4]), we further corrected, refined, and supplemented the constraints. It was observed that Kucukkoc's formulation lacks certain constraints necessary to ensure feasible solutions. Constraint set (13) is therefore included in the new formulation; constraint sets (1) and (7) were replaced with the corresponding sets (10)–(12); and constraints (8) were supplemented with constraint (14) to improve the formulation. It is worth mentioning that constraints (2)–(6) remain unchanged from Kucukkoc's formulation.

$$\text{Minimize } C_{\max} \tag{9}$$

Subject to.
$\sum_{j\in J} X_{ji} = 1; \forall i \in I$ (2).
$\sum_{i\in I} a_i \cdot X_{ji} \le a_m; \forall j \in J$ (3).
$\sum_{i\in I} X_{(j+1)i} \le \psi \cdot \sum_{i\in I} X_{ji}; \forall j \in J\backslash\{j_n\}$ (4).
$C_{(j-1)} + P_j \le C_j; \forall j \in J$ (5).
$C_0 = 0$ (6)

$$C_{\max} \ge C_j; \forall j \in J \tag{10}$$

$$X_{ji} \cdot h_i \le H_j; \forall i \in I; \forall j \in J \tag{11}$$

$$P_j = s_t \cdot Z_j + u_t \cdot \sum_{i\in I} v_i \cdot X_{ji} + h_t \cdot H_j; \quad \forall j \in J \tag{12}$$

$$\sum_{i\in I} X_{ji} \le \psi \cdot Z_j; \forall j \in J \tag{13}$$

$$X_{ji}, \quad Z_j \in \{0,1\}; C_j \ge 0; H_j \ge 0; \forall j \in J; \forall i \in I \tag{14}$$

The objective function (9) aims to minimize makespan. Constraint set (2) restricts every part to be assigned to only one job batch for processing. Constraint set (3) ensures that the total processing area for a batch of parts will not exceed the machine's limitations. Constraint set (4) guarantees that the job batches are set incrementally. This constraint is a symmetry-breaking constraint designed to reduce the state space. Constraint set (5) ensures that the completion time of a job batch is greater than or equal to the sum of the completion time of the preceding job batch and the production time of the current job batch. Constraint set (6) specifies the start processing time of the first job batch as zero. The makespan is bounded in the constraint set (10). Constraint set (11) defines the maximum height of parts processed in each job batch. The processing time of each job batch is calculated using constraint set (12), which includes the setup time, the material molding time required for the total volume of the allocated parts, and the powder build-up time required across all layers of the allocated parts. Constraint set (13) provides a valid formulation, ensuring that $Z_j = 1$ if $\sum_{i\in I} X_{ji} \ge 1$. Without these constraints, Z_j for different j values could be set to zero in a feasible solution. Constraint set (14) defines the ranges of decision variables, as the last constraint in $MILP_{F1}$.

3.2.2. $MILP_{F2}$

To enhance the computational efficiency in solving the $1|batch\{AM\}|C_{\max}$ problem, an alternative MILP formulation with symmetry-breaking constraints is proposed below.

Minimize $C_{\max}$ (9).
Subject to.
$\sum_{j\in J} X_{ji} = 1; \forall i \in I$ (2).
$C_{(j-1)} + P_j \le C_j; \forall j \in J$ (5).
$C_0 = 0$ (6).
$C_{\max} \ge C_j; \forall j \in J$ (10).
$X_{ji} \cdot h_i \le H_j; \forall i \in I; \forall j \in J$ (11)

$$\sum_{i\in I} a_i \cdot X_{ji} \le a_m \cdot Z_j; \quad \forall j \in J \tag{15}$$

$$\sum_{i\in I} X_{ji} \ge Z_j; \forall j \in J \tag{16}$$

$$Z_j \geq Z_{j+1}; \forall j \in J \backslash \{j_n\} \tag{17}$$

$$P_j \geq s_t \cdot Z_j + u_t \cdot \sum_{i \in I} v_i \cdot X_{ji} + h_t \cdot H_j; \quad \forall j \in J \tag{18}$$

$$X_{ji}, \quad Z_j \in \{0,1\}; C_j \geq 0; H_j \geq 0; \forall j \in J; \forall i \in I \tag{19}$$

Constraint set (15) specifies that the cumulated area of parts for an active batch will not exceed the maximum area on the build plate. Constraint set (16) ensures that at least one job is allocated to an active batch. Constraint set (17) guarantees that all active batches are ordered consecutively. Constraint set (18) states that the processing time of each batch must be equal to or greater than the sum of the setup time and the processing time of the jobs assigned to that batch. Finally, constraint set (19) defines the ranges of decision variables in MILP_{F2}.

4. Solution method

Since the $1|batch\{AM\}|C_{\max}$ problem is strongly *NP*-hard [4], the proposed MILP models cannot efficiently solve large-scale instances of the problem. Therefore, an effective and efficient heuristic algorithm is required to find dependable schedules within a reasonable computational time when dealing with industrial-scale problems. This section introduces a beam search-based heuristic algorithm, named Tweaked Iterative Beam Search (TIBS), which is designed to effectively and efficiently address the problem. This section is organized as follows: Section 4.1 presents an overview of the proposed TIBS algorithm. Section 4.2 outlines the computational procedure, explaining each step of the TIBS algorithm to provide a comprehensive understanding of its implementation. Section 4.3 introduces the initialization module, which describes the solution initialization procedure to ensure that the algorithm starts with a feasible solution. Section 4.4 explains the lower bound calculation and node evaluation module, explaining how solutions are evaluated and filtered throughout the search process. Section 4.5 describes the beam width update mechanism, which dynamically adjusts the search scope to balance exploration and computational efficiency.

4.1. Overview

Beam Search is an incomplete tree search algorithm that selects the most promising D nodes at each level for further exploration based on the node evaluation module while discarding other nodes. The beam width controls the search range and computational time. A larger beam width increases the search range and duration, leading to better solutions. As an effective search algorithm, Beam Search [38] has been successfully applied to various scheduling problems, including single-machine [39], no-wait flow-shop [40], permutation flow-shop [41], and unrelated parallel-machine scheduling [42]. Despite its strong record in solving complex mathematical problems, Beam Search has not yet been adopted for AMPS.

Iterative Beam Search Algorithm

Input: root node
$D \leftarrow D_0$
Candidates $\leftarrow$ {root}
while *time limit not reached* **do**
  while *Candidates* $\neq \emptyset$ **do**
    nextLevel $\leftarrow$ *branching Candidates*
    Candidates $\leftarrow$ best D nodes among nextLevel
  end
  $D \leftarrow D \times G$
end
Output: the best node

Fig. 2. Pseudo-code of the IBS algorithm.

Iterative Beam Search (IBS) is an extension of the classic Beam Search Algorithm. As shown in the pseudo-code (see Fig. 2), IBS iteratively performs a beam search until the predefined time limit is reached [43]. Specifically, the algorithm starts with an initial beam width, geometrically increasing the beam width by a factor of G during every iteration, and then exploring the tree using the updated beam width. Throughout the branching process, the node evaluation module is utilized to evaluate the quality of nodes at each level and to select the most promising ones for further branching. In contrast to the branch and bound algorithm, these mechanisms significantly improve the search efficiency of IBS by prohibiting backtracking and focusing solely on the most promising nodes, rather than examining all possible branches.

In this study, we propose an enhanced IBS algorithm, named Tweaked Iterative Beam Search (TIBS), for solving the $1|batch\{AM\}|C_{\max}$ problem. The TIBS adds two novel mechanisms, namely the advanced initialization module and the solution feasibility evaluation mechanism, to enhance the search efficiency. The advanced initialization module can yield an effective initial solution to serve as the baseline for calculating the lower bound, and thus, fathoms the inferior nodes in the early stage. The solution feasibility evaluation mechanism can verify and eliminate infeasible nodes based on the constraints of the $1|batch\{AM\}|C_{\max}$ problem. Note that after fathoming the inferior nodes and eliminating infeasible nodes, the remaining nodes for further branching may be fewer than the beam width at the corresponding level. Therefore, these novel mechanisms can limit memory usage and speed up the search procedure by decreasing the number of branch nodes required to find a high-quality solution. The following notations are employed to define the algorithm. First, the initialization module is introduced, followed by a step-by-step guide for implementing the TIBS algorithm.

Additional notations:

D: Initial beam width.

L: Level; refers to the depth from the root node to the current node.

π_0: Root node.

U: Set of unassigned parts.

$\pi_{L,k}$: Node identity: represents the k^{th} node of layer L (counted from the left-hand side).

$Bound(\pi_{L,k})$: The lower bound of the makespan for the node $\pi_{L,k}$.

$Score(\pi_{L,n})$: Score of the evaluation function for the node $\pi_{L,k}$.

4.2. Computational steps of TIBS

The computational procedure of TIBS consists of the following seven steps. The details of the mechanisms are described in the following subsections, with an illustrative example provided in the Appendix.

Step 1: Generate the initial solution. Apply the initialization module in Section 4.3. The resulting initial solution will serve as the baseline for calculating the upper bound (UB) of the makespan.

Step 2: Set the parameters and normalize the data. Determine the root node π_0, initial beam width D, and normalize the data associated with every part (*i.e.*, a value within interval $[0,1]$).

Step 3: Set $L = 0$ for the initial node, and increment L by one for the subsequent branching layers. Apply the forward branching scheme [43] to branch each candidate node. Construct solutions starting from the root node, which contains no parts; this scheme incrementally builds the solution tree by branching forward from each candidate node to the next stage without reversing or revisiting previous stages. This branching process generates candidate (partial) solutions by inserting each unscheduled part into eligible job batches, forming the next layer of nodes.

Step 4: Verify the feasibility of the nodes. Check if the solutions (*i.e.*, schedules) in the current layer satisfy the constraints and conditions. Delete equivalent nodes for computational efficiency.

Step 5: Apply the lower bound calculation and node evaluation procedure described in Section 4.4 for screening the branching nodes.

Table 2
Specifications of the illustrative example [37].

Part	Height (cm)	Area (cm^2)	Volume (cm^3)
1	35.23	477.08	3638.04
2	1.18	994.67	330.31
3	19.13	438.27	5551.95
4	17.99	1069.27	6847.29
5	13.87	71.15	670.69
6	9.22	120.45	119.78
7	25.64	203.14	851.55
8	34.03	577.83	4844.55
9	15.23	23.28	228.87
10	17.58	104.93	302.03
11	20.89	1009.71	5144.07
12	10.54	57.62	246.42
13	34.61	51.41	295.45
14	29.33	251.00	5053.68
15	12.32	81.40	291.65

First, calculate the lower bound, denoted as $Bound(\pi_{L,n})$, for each node in the current layer, and prune the branching nodes whose lower bounds are higher than or equal to the makespan of the current solution. Then, the best D nodes with the smallest scores, $Score(\pi_{L,k})$, are retained for the next round of branching. In this procedure, competitive solutions will proceed to the next step.

Step 6: Apply the beam width update mechanism described in Section 4.5 to update the beam width value. If there are any unassigned parts, return to Step 3 and update the beam width as $D \leftarrow D \times 2$. Otherwise, proceed to the next step.

Step 7: Stopping condition. If $L = i_n$, then output the best objective function value corresponding to the best solution.

4.3. Initialization module

The solution initialization procedure begins by sorting the parts in descending order based on their heights. Considering the illustrative example in Table 2, the initial sorting results in the following order: {1, 13, 8, 14, 7, 11, 3, 4, 10, 9, 5, 15, 12, 6, 2}. The sorted parts are then sequentially assigned to the production batches based on the well-known First-Fit strategy. As the remaining area of the build plate decreases, a new batch must be added when no additional parts can fit into the available space. This ensures the efficient utilization of the build plate. This process not only determines an UB on the makespan, but also establishes the number of build plates used to avoid excessive branching. Assuming a maximum area of about 1600 cubic centimeters, $\Pi = \{(1, 13, 8, 14, 7, 9), (11, 3, 10), (4, 5, 15, 12, 6), (2)\}$ is the output of the initialization procedure.

4.4. Lower bound calculation and node evaluation module

First, calculate the lower bound, denoted as $Bound(\pi_{L,n})$, for each node in the current layer, and prune the branching nodes whose lower bounds are higher than or equal to the makespan of the current solution. Eq. (20) is used to calculate the lower bound of the makespan. First, the total processing time of all batches is calculated based on the assigned parts. Then, the total processing time of all unassigned parts, $i \in U$, is summed up by considering the time required for molding the material per unit of volume. Note that the maximum number of job batches, j_n, is set equal to the number of job batches in the initial solution to strengthen the calculation of the lower bound.

$$Bound(\pi_{L,k}) = \sum_{j=1}^{j_n} P_j + v_t \cdot \sum_{i \in U} v_i \tag{20}$$

Then, in the node evaluation module, the best D nodes with the smallest scores, $Score(\pi_{L,k})$, are retained for the next round of branching. The score of each candidate node is calculated based on an objective function that uses a weighted sum of the influencing factors. Let H_j^{nor} and V_j^{nor} denote the highest normalized height value of batch j and the total normalized part volume of batch j, respectively. The node score is calculated using Eq. (21), which is a weighted average of the total H_j^{nor} and V_j^{nor}. In this situation, a smaller value of $Score(\pi_{L,k})$ indicates a higher likelihood of obtaining a lower objective function value when filtering the nodes.

$$Score(\pi_{L,k}) = \frac{h_t}{h_t + v_t} \sum_{j=1}^{j_n} H_j^{nor} + \left(1 - \frac{h_t}{h_t + v_t}\right) \sum_{j=1}^{j_n} V_j^{nor} \tag{21}$$

4.5. Beam width update mechanism

The beam width parameter, D, regulates the search procedure by limiting the number of branch nodes. In TIBS, the initial value of D is determined in such a way that only the best nodes are kept in the first layer. As the algorithm progresses to the next layers through branches, the search space can be expanded by setting $D \leftarrow D \times 2$. The detailed parameter settings will be discussed in the fourth chapter. Fig. 3 shows an exemplary search tree with $D = 2$. In this diagram, the nodes highlighted in red represent the nodes that have succeeded in the evaluation procedure.

In this method, all part data needs to be normalized in relation to the beam width. The goal is to scale the part data to the range of 0 to 1 to facilitate the calculation of the node score, as explained in Section 4.5. This is done to assess the extent to which the height or volume influences the value of the objective function and ensure that the measures are proportionate. Normalization can be performed using Eqs. (22) and (23), where h_i and h_i^{nor} represent the current and normalized height values, respectively; while $h_{\min}$ and $h_{\max}$ correspond to the shortest and tallest parts, respectively. The same notation system is used to normalize the part volume.

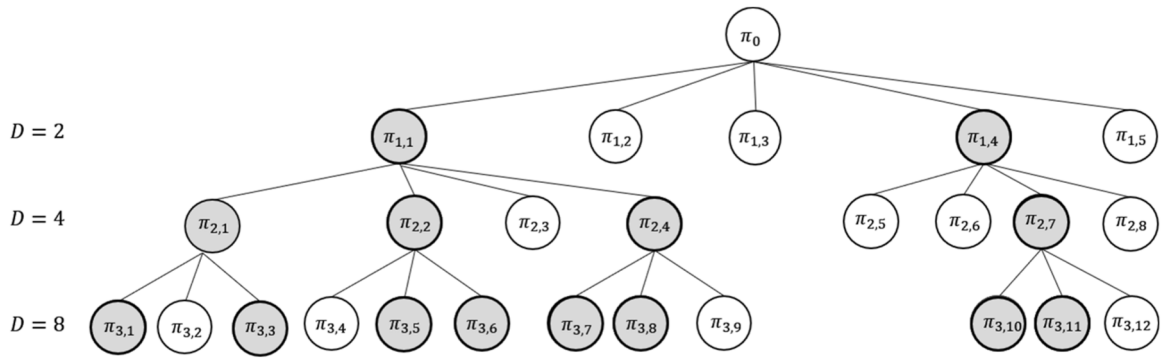

Fig. 3. Illustration of the search tree with $D = 2$.

$$h_i^{nor} = (h_i - h_{\min})/(h_{\max} - h_{\min}) \quad (22)$$

$$v_i^{nor} = (v_i - v_{\min})/(v_{\max} - v_{\min}) \quad (23)$$

5. Numerical analysis

This chapter begins by solving the mathematical optimization problem using exact optimizers for small- and medium-sized instances. A comparison of the developed algorithm and MILP model with the baseline of reference [43] follows. For this purpose, the algorithms are coded and compiled using the Visual Studio 2022 C++ programming language. The implementation was carried out on a personal computer with an Intel(R) Core(TM) i7–8550U CPU running at 1.80 GHz and 16 GB of RAM.

Table 3
Summary of the test instance specifications.

Dataset	Code	Number of parts
Kucukkoc's	P1~ P45	6–46
Small	S1~ S60	10, 20
Medium	M1~ M60	30, 50
Large	L1~L60	100, 150

5.1. Preliminaries

This study utilizes the standard datasets developed by [4,44]; these datasets, generated based on some preliminary work and the author's experience in the AM industry, consist of 45 small- to medium-sized instances, representing real-world scenarios in the best possible way. Furthermore, randomly generated instances of small-, medium-, and large-sized (60 instances of each) are added to create a total of 45 + 180 = 225 distinct test instances for numerical analysis. The ranges of parts in the dataset are systematically determined to provide a diversified set of test problems in various sizes and to ensure the robustness and applicability of the proposed methodology in diverse manufacturing environments. The instance specifications are summarized in Table 3.

Relative Percentage Deviation (*RPD*) is used to compare the algorithms. The formula for calculating *RPD* calculation is shown in Eq. (24), where C_{sol}^{h} represents the makespan obtained by a specific algorithm, and C_{best} refers to the best makespan obtained by all compared

Table 4
Results of solving small- to medium-sized instances by Kucukkoc (best-found solutions in bold).

Instance	Parts	$MILP_{F1}$			$MILP_{F2}$			IBS			TIBS		
		C_{max}	*RPD*	Time	C_{max}	*RPD*	Time	C_{max}	*RPD*	Time	C_{max}	*RPD*	Time
P1	6	201.358 *	0.000	0.04	201.358 *	0.000	0.02	201.358 *	0.000	0.02	201.358 *	0.000	0.01
P2	6	198.831 *	0.000	0.02	198.831 *	0.000	0.02	198.831 *	0.000	0.02	198.831 *	0.000	0.01
P3	7	181.237 *	0.000	0.08	181.237 *	0.000	0.04	181.237 *	0.000	0.09	181.237 *	0.000	0.04
P4	7	173.828 *	0.000	0.02	173.828 *	0.000	0.03	173.828 *	0.000	0.07	173.828 *	0.000	0.02
P5	8	190.959 *	0.000	0.09	190.959 *	0.000	0.04	190.959 *	0.000	0.16	190.959 *	0.000	0.13
P6	8	183.550 *	0.000	0.05	183.550 *	0.000	0.04	183.550 *	0.000	0.13	183.550 *	0.000	0.07
P7	9	266.098 *	0.000	0.19	266.098 *	0.000	0.08	267.127	0.387	0.35	266.098 *	0.000	0.39
P8	9	253.999 *	0.000	0.13	253.999 *	0.000	0.07	253.999 *	0.000	0.22	253.999 *	0.000	0.27
P9	10	283.033 *	0.000	0.32	283.033 *	0.000	0.13	284.363	0.470	0.79	283.033 *	0.000	0.55
P10	10	275.624 *	0.000	0.21	275.624 *	0.000	0.09	275.624 *	0.000	0.71	275.624 *	0.000	0.33
P11	11	374.223 *	0.000	0.81	374.223 *	0.000	0.27	374.223 *	0.000	1.72	374.223 *	0.000	0.58
P12	12	538.089 *	0.000	2.52	538.089 *	0.000	0.82	538.089 *	0.000	3.18	538.089 *	0.000	1.60
P13	12	528.118 *	0.000	0.81	528.118 *	0.000	0.41	528.118 *	0.000	2.36	528.118 *	0.000	1.38
P14	15	393.015 *	0.000	4.73	393.015 *	0.000	0.90	393.015 *	0.000	13.64	393.015 *	0.000	8.44
P15	17	777.154 *	0.000	225.59	777.154 *	0.000	8.17	786.261	1.172	32.21	777.154 *	0.000	23.59
P16	17	794.942 *	0.000	221.65	794.942 *	0.000	5.22	795.692	0.094	28.32	794.942 *	0.000	19.95
P17	21	588.548 *	0.000	155.77	588.548 *	0.000	49.74	605.718	2.917	99.62	588.548 *	0.000	72.87
P18	22	825.887 *	0.000	45.49	825.887 *	0.000	3.89	837.843	1.448	108.76	825.887 *	0.000	74.81
P19	22	865.355 *	0.000	484.56	865.355 *	0.000	107.40	879.765	1.665	130.86	865.355 *	0.000	99.49
P20	23	907.487 *	0.000	3215.06	907.487 *	0.000	56.99	919.697	1.345	151.57	907.487 *	0.000	119.72
P21	25	869.819 *	0.000	250.91	869.819 *	0.000	23.89	875.552	0.659	222.75	869.819 *	0.000	173.95
P22	25	912.207 *	0.000	7200.00	912.207 *	0.000	776.47	924.247	1.320	247.98	912.207 *	0.000	202.11
P23	28	1064.199	0.000	7200.00	1064.199	0.000	7198.36	1085.540	2.005	366.95	1064.199	0.000	298.77
P24	30	1035.510 *	0.000	7200.00	1035.510 *	0.000	2194.72	1072.050	3.529	424.11	1036.730	0.118	280.93
P25	36	1103.264	0.362	7200.00	1099.288	0.000	7200.00	1127.830	2.596	906.94	1099.288	0.000	656.68
P26	36	1134.646	0.783	7200.00	1125.836	0.000	7200.00	1155.430	2.629	888.19	1126.780	0.084	641.29
P27	38	1083.740	0.329	7200.00	1080.265	0.007	7200.00	1114.470	3.174	1095.52	1080.190	0.000	770.76
P28	38	1118.814	0.969	7200.00	1108.074	0.000	7200.00	1157.300	4.442	1119.58	1108.600	0.047	752.56
P29	46	1320.470	1.267	7200.00	1307.215	0.250	7200.00	1366.140	4.769	2298.65	1303.950	0.000	1580.32
P30	46	1359.050	1.522	7200.00	1344.329	0.423	7200.00	1430.110	6.831	2340.98	1338.670	0.000	1675.36
P31	21	579.985 *	0.000	1271.07	579.985 *	0.000	61.27	586.579	1.137	101.67	579.985 *	0.000	81.93
P32	22	855.611 *	0.000	7200.00	855.611 *	0.000	235.90	859.783	0.488	169.79	855.611 *	0.000	139.09
P33	23	907.487 *	0.000	7200.00	907.487 *	0.000	55.35	919.697	1.345	148.63	907.487 *	0.000	119.48
P34	25	900.219	0.069	7200.00	899.596 *	0.000	3091.05	904.279	0.521	272.77	900.345	0.083	228.60
P35	25	912.207 *	0.000	7200.00	912.207 *	0.000	775.66	924.247	1.320	243.96	912.207 *	0.000	202.35
P36	28	1075.560	0.396	7200.00	1071.315	0.000	7200.00	1085.790	1.351	512.3	1071.350	0.003	430.51
P37	28	1091.240	0.547	7200.00	1085.299	0.000	7200.00	1097.250	1.101	466.84	1091.620	0.582	393.44
P38	30	1047.862	0.248	7200.00	1045.272	0.000	7200.00	1072.740	2.628	600.31	1045.380	0.010	462.17
P39	30	1064.450	1.078	7200.00	1053.100	0.000	7200.00	1093.150	3.803	593.85	1059.520	0.610	404.26
P40	36	1140.940	0.308	7200.00	1137.436	0.000	7200.00	1164.980	2.422	1473.4	1140.980	0.312	1250.47
P41	36	1162.700	1.013	7200.00	1152.446	0.122	7200.00	1176.480	2.210	1159.4	1151.040	0.000	878.06
P42	38	1125.560	0.510	7200.00	1119.852	0.000	7200.00	1164.170	3.957	1711.14	1121.520	0.149	1311.08
P43	38	1150.530	2.322	7200.00	1128.284	0.344	7200.00	1191.410	5.958	1240.12	1124.420	0.000	848.54
P44	46	1364.280	1.123	7200.00	1353.854	0.350	7200.00	1437.290	6.535	3404.15	1349.130	0.000	2591.17
P45	46	1395.950	2.315	7200.00	1374.489	0.742	7200.00	1458.710	6.915	2533.94	1364.370	0.000	1901.35
Average			0.337	3810.67		0.050	2885.49		1.848	558.19		0.044	415.54

* : Optimal solutions.

Table 5
Results of solving small-sized instances (best-found solutions in bold).

h	Parts	$MILP_{F1}$			$MILP_{F2}$			IBS			TIBS		
		C_{max}	*RPD*	Time	C_{max}	*RPD*	Time	C_{max}	*RPD*	Time	C_{max}	*RPD*	Time
S1	10	**1169.734** *	0.000	0.149	**1169.734** *	0.000	0.110	1173.020	0.281	0.149	**1169.734** *	0.000	0.353
S2	10	**1851.872** *	0.000	0.099	**1851.872** *	0.000	0.060	**1851.872** *	0.000	0.099	**1851.872** *	0.000	0.321
S3	10	**1557.757** *	0.000	0.087	**1557.757** *	0.000	0.070	**1557.757** *	0.000	0.087	**1557.757** *	0.000	0.759
S4	10	**1885.097** *	0.000	0.102	**1885.097** *	0.000	0.090	**1885.097** *	0.000	0.102	**1885.097** *	0.000	0.384
S5	10	**2089.898** *	0.000	0.124	**2089.898** *	0.000	0.100	2094.780	0.234	0.124	**2089.898** *	0.000	0.401
S6	10	**1255.336** *	0.000	0.094	**1255.336** *	0.000	0.170	**1255.336** *	0.000	0.094	**1255.336** *	0.000	0.233
S7	10	**2699.055** *	0.000	0.168	**2699.055** *	0.000	0.270	**2699.055** *	0.000	0.168	**2699.055** *	0.000	0.276
S8	10	**866.868** *	0.000	0.12	**866.868** *	0.000	0.150	**866.868** *	0.000	0.12	**866.868** *	0.000	0.202
S9	10	**3328.660** *	0.000	0.305	**3328.660** *	0.000	0.210	**3328.660** *	0.000	0.305	**3328.660** *	0.000	0.420
S10	10	**1824.675** *	0.000	0.05	**1824.675** *	0.000	0.100	**1824.675** *	0.000	0.05	**1824.675** *	0.000	0.142
S11	10	**911.096** *	0.000	0.037	**911.096** *	0.000	0.050	**911.096** *	0.000	0.037	**911.096** *	0.000	0.065
S12	10	**4575.461** *	0.000	0.811	**4575.461** *	0.000	0.330	**4575.461** *	0.000	0.811	**4575.461** *	0.000	0.553
S13	10	**2705.052** *	0.000	0.076	**2705.052** *	0.000	0.120	2706.060	0.037	0.076	**2705.052** *	0.000	0.332
S14	10	**3019.863** *	0.000	0.164	**3019.863** *	0.000	0.160	**3019.863** *	0.000	0.164	**3019.863** *	0.000	0.305
S15	10	**2575.298** *	0.000	0.199	**2575.298** *	0.000	0.190	**2575.298** *	0.000	0.199	**2575.298** *	0.000	0.285
S16	10	**668.647** *	0.000	0.764	**668.647** *	0.000	0.280	**668.647** *	0.000	0.764	**668.647** *	0.000	0.371
S17	10	**5036.195** *	0.000	0.558	**5036.195** *	0.000	0.510	**5036.195** *	0.000	0.558	**5036.195** *	0.000	0.425
S18	10	**1130.100** *	0.000	0.173	**1130.100** *	0.000	0.150	**1130.100** *	0.000	0.173	**1130.100** *	0.000	0.221
S19	10	**1162.731** *	0.000	0.125	**1162.731** *	0.000	0.140	1163.670	0.081	0.125	**1162.731** *	0.000	0.237
S20	10	**1261.246** *	0.000	0.173	**1261.246** *	0.000	0.200	**1261.246** *	0.000	0.173	**1261.246** *	0.000	0.248
S21	10	**4086.409** *	0.000	0.198	**4086.409** *	0.000	0.310	**4086.409** *	0.000	0.198	**4086.409** *	0.000	0.313
S22	10	**2292.393** *	0.000	0.131	**2292.393** *	0.000	0.150	**2292.393** *	0.000	0.131	**2292.393** *	0.000	0.246
S23	10	**4247.493** *	0.000	0.133	**4247.493** *	0.000	0.160	**4247.493** *	0.000	0.133	**4247.493** *	0.000	0.240
S24	10	**503.897** *	0.000	0.083	**503.897** *	0.000	0.060	**503.897** *	0.000	0.083	**503.897** *	0.000	0.128
S25	10	**2255.554** *	0.000	0.127	**2255.554** *	0.000	0.070	**2255.554** *	0.000	0.127	**2255.554** *	0.000	0.243
S26	10	**1172.771** *	0.000	0.245	**1172.771** *	0.000	0.220	**1172.771** *	0.000	0.245	**1172.771** *	0.000	0.283
S27	10	**2949.988** *	0.000	0.092	**2949.988** *	0.000	0.140	2958.350	0.283	0.092	2954.850	0.165	0.249
S28	10	**1143.243** *	0.000	0.21	**1143.243** *	0.000	0.180	**1143.243** *	0.000	0.21	**1143.243** *	0.000	0.307
S29	10	**5276.227** *	0.000	0.23	**5276.227** *	0.000	0.130	**5276.227** *	0.000	0.23	**5276.227** *	0.000	0.466
S30	10	**1272.794** *	0.000	0.16	**1272.794** *	0.000	0.150	1273.500	0.056	0.16	**1272.794** *	0.000	0.469
S31	20	**4311.960** *	0.000	136.391	**4311.960** *	0.000	38.820	4338.550	0.617	136.391	**4311.960** *	0.000	71.182
S32	20	**5727.716** *	0.000	948.292	**5727.716** *	0.000	19.820	**5727.716** *	0.000	948.292	**5727.716** *	0.000	88.012
S33	20	**6671.411** *	0.000	7207.49	**6671.411** *	0.000	2729.810	6690.160	0.281	7207.49	6681.570	0.152	110.676
S34	20	**7272.308** *	0.000	7204.495	**7272.308** *	0.000	5994.940	7294.540	0.306	7204.495	**7272.308** *	0.000	96.304
S35	20	**4193.438**	0.000	7201.087	**4193.438**	0.000	7201.510	4200.170	0.161	7201.087	4200.170	0.161	120.733
S36	20	**2900.598** *	0.000	1576.608	**2900.598** *	0.000	100.850	**2900.598** *	0.000	1576.608	**2900.598** *	0.000	127.918
S37	20	**5631.460** *	0.000	2982.862	**5631.460** *	0.000	716.150	5657.910	0.470	2982.862	5631.810	0.006	73.483
S38	20	**1838.104** *	0.000	701.66	**1838.104** *	0.000	23.110	1842.620	0.246	701.66	**1838.104** *	0.000	92.089
S39	20	**2867.030** *	0.000	2545.165	**2867.030** *	0.000	146.890	2876.160	0.318	2545.165	2867.310	0.010	97.497
S40	20	**3659.604** *	0.000	154.537	**3659.604** *	0.000	485.460	3665.710	0.167	154.537	3662.720	0.085	101.102
S41	20	7359.557	0.000	7200.28	**7359.524** *	0.000	6558.550	7378.220	0.254	7200.28	7369.640	0.137	127.994
S42	20	**6147.603** *	0.000	1477.132	**6147.603** *	0.000	36.030	6159.630	0.196	1477.132	**6147.603** *	0.000	85.536
S43	20	**2758.417** *	0.000	1246.637	**2758.417** *	0.000	44.270	2763.610	0.188	1246.637	2762.250	0.139	94.562
S44	20	**5721.425** *	0.000	377.571	**5721.425** *	0.000	17.540	5750.700	0.512	377.571	**5721.425** *	0.000	79.221
S45	20	**2844.237** *	0.000	132.438	**2844.237** *	0.000	16.630	2858.380	0.497	132.438	**2844.237** *	0.000	75.426
S46	20	**4819.011** *	0.000	332.766	**4819.011** *	0.000	79.870	4833.240	0.295	332.766	4822.830	0.079	79.173
S47	20	**6004.091** *	0.000	3903.697	**6004.091** *	0.000	633.950	6015.280	0.186	3903.697	**6004.091** *	0.000	79.284
S48	20	**5117.154** *	0.000	248.335	**5117.154** *	0.000	40.140	5137.230	0.392	248.335	**5117.154** *	0.000	67.709
S49	20	**3016.070** *	0.000	1589.513	**3016.070** *	0.000	158.360	3019.810	0.124	1589.513	3018.050	0.066	107.715
S50	20	**4025.038** *	0.000	5527.874	**4025.038** *	0.000	502.040	4038.900	0.344	5527.874	4040.580	0.386	91.257
S51	20	**2868.070** *	0.000	1479.278	**2868.070** *	0.000	37.790	2874.180	0.213	1479.278	2868.770	0.024	90.912
S52	20	**7588.666** *	0.000	309.327	**7588.666** *	0.000	41.950	7608.770	0.265	309.327	7607.650	0.250	122.696
S53	20	**7515.740** *	0.000	2352.928	**7515.740** *	0.000	91.880	7529.670	0.185	2352.928	**7515.740** *	0.000	85.754
S54	20	**2007.191** *	0.000	161.122	**2007.191** *	0.000	21.860	2012.260	0.253	161.122	2010.370	0.158	112.849
S55	20	**1844.789** *	0.000	1955.873	**1844.789** *	0.000	162.750	1849.180	0.238	1955.873	**1844.789** *	0.000	83.691
S56	20	**1726.294** *	0.000	573.067	**1726.294** *	0.000	12.630	1735.200	0.516	573.067	1728.420	0.123	94.963
S57	20	**7006.269** *	0.000	906.504	**7006.269** *	0.000	301.730	7052.520	0.660	906.504	**7006.269** *	0.000	107.867
S58	20	**4442.645** *	0.000	386.676	**4442.645** *	0.000	11.420	4462.430	0.445	386.676	**4442.645** *	0.000	53.168
S59	20	**2946.012** *	0.000	766.549	**2946.012** *	0.000	38.930	2954.950	0.303	766.549	2948.990	0.101	82.527
S60	20	**4184.128** *	0.000	345.228	**4184.128** *	0.000	56.090	4210.620	0.633	345.228	**4184.128** *	0.000	67.843
Average			0.000	1032.289		0.000	438.780		0.171	41.234		0.034	46.310

*Optimal solutions.

algorithms. In this setup, a smaller *RPD* indicates that the solution is more competitive.

$$RPD = \frac{C^h_{sol} - C_{best}}{C_{best}} \times 100\% \quad (24)$$

5.2. Results

As a first step in the numerical experiments, the small- and medium-sized instances, namely P1-P45, S1-S60, and M1-M60 are considered. The best-found solutions obtained by the compared algorithms and the Gurobi optimizer, along with their respective *RPD*s, are presented in Tables 4–6, respectively. The Gurobi optimizer considers a maximum computational time limit of 7200 s. For the sake of efficiency, we set the big-M parameter, ψ, to 99,999 in this study. The optimal solutions are marked with an asterisk (*), while the best-found solutions are highlighted in **bold** font.

Table 6
Results of solving medium-sized instances (best-found solutions in bold).

Instance	Parts	$MILP_{F1}$			$MILP_{F2}$			IBS			TIBS		
		C_{max}	*RPD*	Time	C_{max}	*RPD*	Time	C_{max}	*RPD*	Time	C_{max}	*RPD*	Time
M1	30	8189.242	0.032	7200.0	**8186.638**	0.000	7200.0	8218.770	0.392	588.0	8205.750	0.233	752.6
M2	30	**6400.28**9	0.000	7200.0	**6400.289**	0.000	7200.0	6443.250	0.671	429.3	6404.240	0.062	525.7
M3	30	9195.558	0.096	7200.0	**9186.766**	0.000	7200.0	9222.610	0.390	553.0	9196.400	0.105	714.9
M4	30	**2423.876**	0.000	7200.0	**2423.876**	0.000	7200.0	2434.530	0.440	654.8	2431.170	0.301	845.9
M5	30	**7181.834**	0.000	7200.0	**7181.834**	0.000	7200.0	7210.720	0.402	513.0	7201.850	0.279	658.6
M6	30	3986.973	0.034	7200.0	**3985.629**	0.000	7200.0	4003.690	0.453	618.9	3987.370	0.044	780.9
M7	30	4082.800	0.186	7200.0	**4075.228**	0.000	7200.0	4098.840	0.579	752.5	4082.420	0.176	904.0
M8	30	3808.538	0.060	7200.0	**3806.249**	0.000	7200.0	3821.890	0.411	641.7	3809.450	0.084	846.8
M9	30	**4071.971**	0.000	7200.0	**4071.971**	0.000	7200.0	4104.170	0.791	615.8	4072.490	0.013	802.8
M10	30	7799.787	0.136	7200.0	**7789.161**	0.000	7200.0	7817.470	0.363	524.6	7796.570	0.095	687.3
M11	30	**9078.816** *	0.000	7200.0	**9078.816** *	0.000	7200.0	9123.060	0.487	606.2	**9078.816** *	0.000	790.5
M12	30	**4165.689**	0.000	7200.0	**4165.689**	0.000	7200.0	4185.460	0.475	595.7	4171.420	0.138	800.2
M13	30	9292.206	0.083	7200.0	**9284.530**	0.000	7200.0	9343.530	0.635	582.4	9293.130	0.093	761.7
M14	30	**9657.076**	0.000	7200.0	**9657.076**	0.000	7200.0	9713.930	0.589	517.2	**9657.076**	0.000	693.8
M15	30	5051.489	0.037	7200.0	**5049.634**	0.000	7200.0	5056.750	0.141	526.9	5050.290	0.013	716.8
M16	30	8157.077	0.083	7200.0	**8150.315**	0.000	7200.0	8193.490	0.530	521.8	8169.060	0.230	708.7
M17	30	2365.821	0.005	7200.0	**2365.702**	0.000	7200.0	2384.250	0.784	549.7	2368.960	0.138	721.7
M18	30	8430.841	0.027	7200.0	**8428.531**	0.000	7200.0	8469.480	0.486	641.0	8440.540	0.142	850.2
M19	30	5914.207	0.089	7200.0	**5908.971**	0.000	7200.0	5955.480	0.787	566.9	5912.130	0.053	717.4
M20	30	6860.207	0.096	7200.0	**6853.617**	0.000	7200.0	6866.660	0.190	725.3	6859.120	0.080	959.0
M21	30	3848.470	0.185	7200.0	**3841.379**	0.000	7200.0	3875.930	0.899	688.8	3847.010	0.147	886.2
M22	30	**8748.528**	0.000	7200.0	**8748.528**	0.000	7200.0	8814.350	0.752	556.6	8778.110	0.338	732.5
M23	30	12311.561	0.097	7200.0	**12299.675**	0.000	7200.0	12360.500	0.495	734.7	12320.700	0.171	892.8
M24	30	2901.885	0.067	7200.0	**2899.946**	0.000	7200.0	2924.590	0.850	634.9	2904.780	0.167	844.2
M25	30	3177.891	0.039	7200.0	**3176.652**	0.000	7200.0	3204.160	0.866	571.2	**3176.652**	0.000	776.6
M26	30	3071.099	0.276	7200.0	**3062.643**	0.000	7200.0	3083.290	0.674	536.5	3063.870	0.040	699.3
M27	30	**6085.712** *	0.000	7200.0	**6085.712** *	0.000	7200.0	6116.460	0.505	532.5	6090.230	0.074	704.8
M28	30	2014.489	0.196	7200.0	**2010.542**	0.000	7200.0	2032.770	1.106	480.3	2012.880	0.116	628.7
M29	30	2990.858	0.007	7200.0	**2990.662**	0.000	7200.0	3026.930	1.213	613.6	2993.250	0.087	727.8
M30	30	**8878.095**	0.000	7200.0	**8878.095**	0.000	7200.0	8928.560	0.568	567.2	8892.430	0.161	755.5
M31	50	10732.765	0.568	7200.0	10705.885	0.317	7200.0	10817.300	1.361	2085.0	**10672.100**	0.000	507.0
M32	50	7721.216	0.109	7200.0	**7712.802**	0.000	7200.0	7777.030	0.833	2520.9	7716.220	0.044	517.9
M33	50	16639.389	0.132	7200.0	16621.623	0.025	7200.0	16728.400	0.667	2688.5	**16617.500**	0.000	665.9
M34	50	16344.914	0.176	7200.0	16366.026	0.305	7200.0	16459.300	0.877	2522.1	**16316.200**	0.000	614.1
M35	50	6819.792	0.280	7200.0	6805.038	0.063	7200.0	6858.330	0.846	4149.5	**6800.780**	0.000	614.7
M36	50	19240.029	0.367	7200.0	19198.427	0.150	7200.0	19262.500	0.485	1313.1	**19169.600**	0.000	642.7
M37	50	12640.095	0.380	7200.0	12631.639	0.312	7200.0	12671.900	0.632	402.0	**12592.300**	0.000	492.8
M38	50	16285.462	0.324	7200.0	16272.896	0.246	7200.0	16318.800	0.529	12590.9	**16232.900**	0.000	527.9
M39	50	12495.358	0.559	7200.0	**12425.872**	0.000	7200.0	12590.900	1.328	411.2	12437.600	0.094	506.3
M40	50	14103.086	0.399	7200.0	14070.158	0.165	7200.0	14116.100	0.492	382.9	**14047.000**	0.000	474.8
M41	50	4891.536	0.212	7200.0	4890.168	0.184	7200.0	4961.390	1.643	511.9	**4881.180**	0.000	641.8
M42	50	5635.408	0.579	7200.0	5617.966	0.268	7200.0	5639.970	0.661	328.2	**5602.940**	0.000	420.2
M43	50	15334.928	0.624	7200.0	15268.744	0.189	7200.0	15348.900	0.715	355.2	**15239.900**	0.000	451.9
M44	50	15579.268	0.299	7200.0	**15532.894**	0.000	7200.0	15595.500	0.403	485.2	15535.300	0.015	601.6
M45	50	4505.984	0.643	7200.0	4482.086	0.109	7200.0	4507.950	0.687	337.8	**4477.200**	0.000	417.3
M46	50	7005.234	0.170	7200.0	7008.083	0.211	7200.0	7025.320	0.457	483.0	**6993.350**	0.000	596.6
M47	50	5817.325	0.312	7200.0	5816.828	0.303	7200.0	5862.250	1.087	575.0	**5799.240**	0.000	693.3
M48	50	11198.146	0.098	7200.0	**11187.212**	0.000	7200.0	11270.800	0.747	480.5	11192.500	0.047	592.7
M49	50	4728.151	0.873	7200.0	4699.459	0.261	7200.0	4709.700	0.479	368.6	**4687.230**	0.000	456.5
M50	50	5087.472	0.454	7200.0	5074.460	0.197	7200.0	5120.490	1.106	511.3	**5064.460**	0.000	629.8
M51	50	6194.211	0.509	7200.0	6176.410	0.220	7200.0	6217.460	0.886	368.0	**6162.850**	0.000	455.0
M52	50	6700.427	0.243	7200.0	6688.912	0.070	7200.0	6731.910	0.714	501.5	**6684.210**	0.000	618.1
M53	50	13857.440	0.448	7200.0	13820.480	0.180	7200.0	13915.700	0.871	430.7	**13795.600**	0.000	523.6
M54	50	13141.237	0.151	7200.0	**13121.385**	0.000	7200.0	13189.500	0.519	546.5	13122.300	0.007	671.0
M55	50	12994.412	0.452	7200.0	12984.470	0.375	7200.0	13018.900	0.641	486.1	**12936.000**	0.000	592.6
M56	50	5971.248	0.378	7200.0	5965.332	0.278	7200.0	5990.190	0.696	512.7	**5948.770**	0.000	633.5
M57	50	13913.924	0.126	7200.0	**13896.378**	0.000	7200.0	13962.100	0.473	440.2	13904.700	0.060	542.4
M58	50	17028.859	0.141	7200.0	17029.083	0.143	7200.0	17220.600	1.269	592.1	**17004.800**	0.000	720.2
M59	50	8307.253	0.417	7200.0	8288.759	0.194	7200.0	8326.960	0.656	518.8	**8272.720**	0.000	642.0
M60	50	18820.422	0.167	7200.0	18802.446	0.072	7200.0	18851.800	0.334	494.0	**18789.000**	0.000	611.4
Average			0.207	7200.0		0.081	7200.0		0.684	933.9		0.064	666.1

* : Optimal solutions.

Table 7
Optimality and best-found solution count for small- to medium-sized instances.

Method	Optimal solutions	Best-found solutions
$MILP_{F1}$	87	97
$MILP_{F2}$	89	133
IBS	38	38
TIBS	71	106

The numerical analysis considering the small- and medium-sized instances reveals that TIBS records an average *RPD* of 0.044, compared to 1.848 for IBS, 0.337 for the first MILP formulation ($MILP_{F1}$), and 0.050 for the second MILP formulation ($MILP_{F2}$). The first observation is that $MILP_{F2}$ resulted in more competitive solutions and shorter CPU times compared to $MILP_{F1}$. Besides, the proposed $MILP_{F2}$ enabled the commercial solver to solve some of the medium-sized instances that could not be solved with the baseline formulation.

Table 8
Results of solving large-sized instances using the MILP methods (7200 s).

Instance	Parts	$MILP_{F1}$			$MILP_{F2}$			IBS			TIBS		
		C_{max}	*RPD*	Time	C_{max}	*RPD*	Time	C_{max}	*RPD*	Time	C_{max}	*PRD*	Time
L1	100	27295.911	4.581	7200	26339.737	0.917	7200	26279.700	0.687	7200	26100.300	0.000	7200
L2	100	—	100.000	7200	30981.086	0.513	7200	—	100.000	7200	30823.100	0.000	7200
L3	100	—	100.000	7200	11722.261	0.773	7200	—	100.000	7200	11632.400	0.000	7200
L4	100	—	100.000	7200	25954.742	0.452	7200	—	100.000	7200	25838.000	0.000	7200
L5	100	—	100.000	7200	12247.773	0.740	7200	—	100.000	7200	12157.800	0.000	7200
L6	100	—	100.000	7200	21988.952	0.568	7200	—	100.000	7200	21864.700	0.000	7200
L7	100	32658.102	2.557	7200	31974.460	0.410	7200	—	100.000	7200	31843.800	0.000	7200
L8	100	—	100.000	7200	26440.043	1.030	7200	—	100.000	7200	26170.600	0.000	7200
L9	100	—	100.000	7200	26226.303	0.578	7200	—	100.000	7200	26075.600	0.000	7200
L10	100	—	100.000	7200	21497.665	0.641	7200	—	100.000	7200	21360.800	0.000	7200
L11	100	—	100.000	7200	24267.437	0.792	7200	—	100.000	7200	24076.800	0.000	7200
L12	100	12944.930	4.096	7200	12483.539	0.385	7200	—	100.000	7200	12435.600	0.000	7200
L13	100	24580.673	3.870	7200	23839.039	0.736	7200	—	100.000	7200	23664.900	0.000	7200
L14	100	31461.758	2.291	7200	30866.646	12.428	7200	—	100.000	7200	30757.200	12.030	7200
L15	100	—	100.000	7200	27933.528	0.578	7200	—	100.000	7200	27772.900	0.000	7200
L16	100	—	100.000	7200	13576.928	0.403	7200	—	100.000	7200	13522.500	0.000	7200
L17	100	—	100.000	7200	28867.556	0.612	7200	—	100.000	7200	28692.100	0.000	7200
L18	100	—	100.000	7200	30336.304	0.702	7200	—	100.000	7200	30124.800	0.000	7200
L19	100	—	100.000	7200	28376.469	0.416	7200	—	100.000	7200	28258.900	0.000	7200
L20	100	33502.762	2.459	7200	32964.730	0.813	7200	—	100.000	7200	32698.800	0.000	7200
L21	100	—	100.000	7200	25779.072	0.510	7200	—	100.000	7200	25648.300	0.000	7200
L22	100	—	100.000	7200	11406.425	0.504	7200	—	100.000	7200	11349.200	0.000	7200
L23	100	—	100.000	7200	23963.389	0.998	7200	—	100.000	7200	23726.500	0.000	7200
L24	100	—	100.000	7200	10903.350	0.511	7200	—	100.000	7200	10847.900	0.000	7200
L25	100	—	100.000	7200	28727.345	0.431	7200	—	100.000	7200	28604.000	0.000	7200
L26	100	—	100.000	7200	23164.954	1.167	7200	—	100.000	7200	22897.700	0.000	7200
L27	100	—	100.000	7200	25520.664	0.543	7200	—	100.000	7200	25382.900	0.000	7200
L28	100	—	100.000	7200	27741.517	0.563	7200	—	100.000	7200	27586.100	0.000	7200
L29	100	—	100.000	7200	26585.279	0.438	7200	—	100.000	7200	26469.300	0.000	7200
L30	100	—	100.000	7200	26855.152	0.554	7200	—	100.000	7200	26707.300	0.000	7200
L31	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	41770.500	0.000	7200
L32	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	36534.600	0.000	7200
L33	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	38734.300	0.000	7200
L34	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	39307.000	0.000	7200
L35	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	19159.400	0.000	7200
L36	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	42337.500	0.000	7200
L37	150	—	100.000	7200	19057.814	1.087	7200	—	100.000	7200	18852.900	0.000	7200
L38	150	—	100.000	7200	16809.176	1.079	7200	—	100.000	7200	16629.800	0.000	7200
L39	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	46478.400	0.000	7200
L40	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	42861.400	0.000	7200
L41	150	—	100.000	7200	16814.399	1.061	7200	—	100.000	7200	16637.800	0.000	7200
L42	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	17650.000	0.000	7200
L43	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	37487.700	0.000	7200
L44	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	35462.100	0.000	7200
L45	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	41676.300	0.000	7200
L46	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	18211.700	0.000	7200
L47	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	19574.100	0.000	7200
L48	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	44099.800	0.000	7200
L49	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	39907.900	0.000	7200
L50	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	43683.700	0.000	7200
L51	150	—	100.000	7200	19712.848	1.062	7200	—	100.000	7200	19505.700	0.000	7200
L52	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	15113.400	0.000	7200
L53	150	—	100.000	7200	46641.656	1.597	7200	—	100.000	7200	45908.600	0.000	7200
L54	150	—	100.000	7200	21113.806	0.432	7200	—	100.000	7200	21023.000	0.000	7200
L55	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	18264.300	0.000	7200
L56	150	—	100.000	7200	19833.631	0.630	7200	—	100.000	7200	19709.400	0.000	7200
L57	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	41738.200	0.000	7200
L58	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	42085.000	0.000	7200
L59	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	34705.500	0.000	7200
L60	150	—	100.000	7200	—	100.000	7200	—	100.000	7200	42965.400	0.000	7200
Average			90.331	7200		38.760	7200		98.345	7200		0.000	7200

It is also noted that TIBS yielded more competitive solutions than IBS in all of the medium-sized instances and also outperformed MILP-based methods. The experiments showed that starting with a better initial solution reduces the subsequent computations and hence the average CPU time decreases compared to the baseline.

Table 7 compares the number of optimal solutions and best solutions obtained by different solution approaches considering ***165*** small- to medium-sized instances. The exact solver could yield more optimal and best-found solutions using $MILP_{F2}$ than the baseline MILP formulation (*i.e.*, $MILP_{F1}$). Likewise, TIBS yielded more optimal and best-found solutions than the baseline IBS algorithm. The proposed improvement in the solution algorithm reduced the gap with the commercial software. Overall, TIBS yielded fewer optimal solutions than $MILP_{F2}$ but required substantially shorter computational time.

The experimental results of the 60 large-sized instances considering a maximum computational time of 7200 s are listed in Table 8. The commercial solver could yield a feasible solution in only 6 out of the 60 instances while using the corrected baseline MILP formulation, $MILP_{F1}$.

Table 9
Results of solving large-sized instances using the Iterative Beam Search algorithms (unlimited computational time).

Instance	Parts	IBS			TIBS		
		C_{max}	RPD	Time	C_{max}	RPD	Time
L1	100	26279.7	0.69%	6926.74	26100.3	0.00%	7380.73
L2	100	31148.2	1.05%	23517.50	30823.1	0.00%	30175.20
L3	100	11753.6	1.04%	43250.50	11632.4	0.00%	6959.71
L4	100	26139.2	1.17%	8668.45	25838.0	0.00%	9081.82
L5	100	12355.6	1.63%	9451.92	12157.8	0.00%	9563.16
L6	100	22173.8	1.41%	7350.55	21864.7	0.00%	6497.02
L7	100	32234.4	1.23%	22507.90	31843.8	0.00%	10797.90
L8	100	26456.8	1.09%	7223.02	26170.6	0.00%	7253.10
L9	100	26480.7	1.55%	10111.00	26075.9	0.00%	8374.75
L10	100	21694.9	1.56%	8743.37	21360.8	0.00%	7226.99
L11	100	24414.8	1.40%	9785.45	24076.8	0.00%	8136.77
L12	100	12536.9	0.81%	10031.70	12435.6	0.00%	8859.14
L13	100	23857.9	0.82%	7800.30	23664.9	0.00%	6959.14
L14	100	31104.4	1.13%	9562.32	30757.2	0.00%	8197.20
L15	100	28264.3	1.77%	11495.50	27772.9	0.00%	9216.74
L16	100	13675.7	1.13%	11234.20	13522.5	0.00%	9428.13
L17	100	28870.0	0.62%	9174.37	28692.1	0.00%	7630.68
L18	100	30328.9	0.68%	10123.50	30124.8	0.00%	8356.88
L19	100	28467.2	0.74%	8270.30	28258.9	0.00%	6855.16
L20	100	33172.8	1.45%	8765.17	32698.8	0.00%	6991.40
L21	100	26055.2	1.59%	10359.70	25648.3	0.00%	8020.86
L22	100	11454.4	0.93%	9386.44	11349.2	0.00%	7819.40
L23	100	24019.9	1.24%	10834.80	23726.5	0.00%	8906.15
L24	100	10960.0	1.03%	7671.10	10847.9	0.00%	6371.84
L25	100	29031.2	1.49%	10801.20	28604.0	0.00%	8764.20
L26	100	23147.2	1.09%	10320.00	22897.7	0.00%	8013.60
L27	100	25649.6	1.05%	12385.00	25382.9	0.00%	10567.30
L28	100	27968.6	1.39%	9463.59	27586.1	0.00%	7544.80
L29	100	26760.7	1.10%	11079.50	26469.3	0.00%	9163.20
L30	100	26974.8	1.00%	10694.00	26707.3	0.00%	8455.40
L31	150	42521.5	1.80%	50884.00	41770.5	0.00%	47663.80
L32	150	36828.0	0.80%	42938.60	36534.6	0.00%	40134.50
L33	150	39137.4	1.04%	34373.20	38734.3	0.00%	32354.10
L34	150	39750.6	1.13%	47519.40	39307.0	0.00%	45511.30
L35	150	19529.9	1.93%	61840.40	19159.4	0.00%	58143.50
L36	150	42977.5	1.51%	48018.60	42337.5	0.00%	46132.40
L37	150	19094.9	1.28%	34586.30	18852.9	0.00%	33541.70
L38	150	16930.7	1.81%	41941.70	16629.8	0.00%	39912.80
L39	150	47099.3	1.34%	51310.40	46478.4	0.00%	48846.50
L40	150	43268.1	0.95%	47574.20	42861.4	0.00%	45561.30
L41	150	16819.0	1.09%	30145.60	16637.8	0.00%	28443.60
L42	150	17967.9	1.80%	51352.00	17650.0	0.00%	67943.50
L43	150	38010.6	1.39%	52379.60	37487.7	0.00%	71489.70
L44	150	36163.1	1.98%	45916.70	35462.1	0.00%	65005.30
L45	150	42077.1	0.96%	41539.90	41676.3	0.00%	57791.90
L46	150	18459.5	1.36%	52988.80	18211.7	0.00%	73385.30
L47	150	19865.3	1.49%	46603.80	19574.1	0.00%	77152.70
L48	150	44595.0	1.12%	38305.20	44099.8	0.00%	65764.50
L49	150	40312.2	1.01%	23113.80	39907.9	0.00%	37102.80
L50	150	44397.8	1.63%	41110.80	43683.7	0.00%	65991.70
L51	150	19771.7	1.36%	37725.50	19505.7	0.00%	74974.80
L52	150	15446.8	2.21%	76381.00	15113.4	0.00%	199133.00
L53	150	46170.9	0.57%	52213.10	45908.6	0.00%	9640.39
L54	150	21191.2	0.80%	54752.40	21023.0	0.00%	11101.70
L55	150	18620.0	1.95%	85568.10	18264.3	0.00%	82486.20
L56	150	19956.8	1.26%	59432.70	19709.4	0.00%	57146.80
L57	150	42086.3	0.83%	24104.50	41738.2	0.00%	21983.70
L58	150	42586.3	1.19%	54432.50	42085.0	0.00%	52013.40
L59	150	35094.2	1.12%	41164.00	34705.5	0.00%	39047.80
L60	150	43339.0	0.87%	44350.70	42965.4	0.00%	42263.40

This number increased to 37 with the proposed formulation, $MILP_{F2}$. The IBS algorithm could obtain a feasible solution to only one instance considering the maximum computational time limit while TIBS could yield the best solution in all of the 60 instances.

To further evaluate the performances of the IBS and TIBS algorithms, the experiments are repeated without time limit. Table 9 confirms that all of the best-found solutions for the large-sized instances were obtained by TIBS although requiring a longer computational time. Considering that the tactical/periodic nature of production scheduling places more emphasis on solution quality than on computational time, one can claim that TIBS is preferred for solving the studied AMSPs.

5.3. Statistical tests

The two-tailed Wilcoxon signed-rank test is used to check whether there is a significant difference between the paired samples. Table 10 provides the results, where the null hypothesis (the solution approach in the row outperforms that in the column) can be rejected in three out of 24 cases. Overall, and supported by the statistical results, it can be concluded that IBS underperforms relative to $MILP_{F1}$, $MILP_{F2}$, and TIBS;

Table 10
Wilcoxon signed-rank test for paired samples of *RPD*.

Small instances				
	RPD_{IBS}	RPD_{F1}	RPD_{F2}	RPD_{TIBS}
RPD_{IBS}	–	$W-value=0.0 \quad p-value=1.140e-07$*	$W-value=0.0 \quad p-value=1.681e-07$*	$W-value=8.0$ $p-value=4.946e-07$*
RPD_{F1}	–	–	$W=6.0 \quad p-value=0.686$	$W=0.0 \quad p-value=1.822e-0.5$*
RPD_{F2}	–	–	–	$W=0.0 \quad p-value=2.702e-0.5$*
RPD_{TIBS}	–	–	–	–
Medium instances				
	RPD_{IBS}	RPD_{F1}	RPD_{F2}	RPD_{TIBS}
RPD_{IBS}	–	$W-value=24 \quad p-value=5.410e-11$*	$W-value=0 \quad p-value=1.630e-11$*	$W-value=0 \quad p-value=1.630e-11$*
RPD_{F1}	–	–	$W-value=44 \quad p-value=6.546e-0$*	$W-value=432 \quad p-value=0.00038$ *
RPD_{F2}	–	–	–	$W-value=850 \quad p-value=0.63229$
RPD_{TIBS}	–	–	–	–
Large instances				
	RPD_{IBS}	RPD_{F1}	RPD_{F2}	RPD_{TIBS}
RPD_{IBS}	–	$W-value=7.0 \quad p-value=0.436$	$W-value=1.0 \quad p-value=1.234-07$*	$W-value=0.0 \quad p-value=1.527e-14$*
RPD_{F1}	–	–	$W-value=0.0 \quad p-value=1.136e-07$*	$W-value=0.0 \quad p-value=1.101e-13$*
RPD_{F2}	–	–	–	$W-value=0.0$ $p-value=1.179e-11$*
RPD_{TIBS}	–	–	–	–
All instances				
	RPD_{IBS}	RPD_{F1}	RPD_{F2}	RPD_{TIBS}
RPD_{IBS}	–	$W-value=254 \quad p-value=1.528e-15$*	$W-value=17.0 \quad p-value=2.104e-23$*	$W-value=8.0 \quad p-value=1.801e-27$*
RPD_{F1}	–	–	$W-value=65 \quad p-value=4.476e-16$*	$W-value=1419.0 \quad p-value=2.343e-14$*
RPD_{F2}	–	–	–	$W-value=2046.0 \quad p-value=3.992e-10$*
RPD_{TIBS}	–	–	–	–

* there exists a significant difference.

$MILP_{F1}$ performs worse than $MILP_{F2}$ and TIBS; and $MILP_{F2}$ is outperformed by TIBS. That is, TIBS is superior to the other three solution methods in terms of overall performance. The difference between the two MILP formulations is not meaningful when considering small instances, while $MILP_{F2}$ performs significantly better than $MILP_{F1}$ in larger instances. Finally, it is observed that IBS and $MILP_{F1}$ perform similarly in large instances.

5.4. Practical implications

There is a consensus amongst practitioners on the positive impacts of producing metal parts using AM; shifting away from traditional manufacturing, however, requires both a reduction in AM processing times and machinery cost. Scheduling the operations considering the available time and resources reduces processing times at the factory level and hence is a prerequisite for a widespread adoption of metal AM.

Decisions on assigning different parts to a batch, the completion time of the batch, and the maximal height of parts in the batch are interrelated, and simultaneously considering them in short-term production planning is necessary. This increases mathematical complexity and hence this study focused on enhancing computational efficiency by narrowing the search space with the help of new mathematical constraints and a node evaluation module in the solution algorithm that helps in filtering the less-promising branching nodes.

Given an aggregate plan, the available time, and resources, production managers should determine the short-term production plans. Such decisions have to be made regularly and hence computer software is required to aid the production managers. In this situation, efficiency and effectiveness become equally important. We showed that the developed algorithm, TIBS is computationally more efficient than the commercial solver while improving the solution quality in large-scale problems. In addition to this, it is observed that TIBS yields significantly better solutions than IBS.

6. Conclusions

This study contributed to the widespread transition to AM-based production by developing an optimization framework for AMPS. The problem studied in this research considers factors such as build plate limitations, machine height, setup time, and parts batching. We have extended two mathematical formulations and developed an iterative version of the BS algorithm for optimizing the problem. In addition, a novel initialization module has been incorporated into the optimization algorithm, along with a new function evaluation method to regulate the node elimination strategy in the beam search algorithms. The evaluation module effectively screens the nodes, reducing computational time while ensuring the best tradeoff between solution quality and speed.

The most notable observations are as follows. TIBS demonstrated superior computational efficiency compared to the MILP solver while producing comparatively good solutions for most small- and medium-sized instances and more competitive solutions when solving large-sized instances. The proposed extension to the IBS algorithm proved to be a breakthrough, as the experimental results showed that it performed equally or better than the IBS algorithm in solving all of the test instances. Statistical tests confirmed this finding. Overall, the experiments confirmed that TIBS is a better alternative for decision support in practice.

This study can be extended in the following directions. The first suggestion for future research is to integrate the AMPS problem with order management, allowing the scheduling of orders from different agents without making assumptions. In addition, studying the scheduling of post-processing activities is worthwhile. Second, our model does not account for the feedstock type or the similarities between additively manufactured parts. In future studies, printing time could be calculated by considering the type of raw materials and the order-dependent setup time between parts. In practice, accounting for both the shape and orientation of parts on the platform would ensure a more accurate fit assessment, especially for parts with similar surface areas but distinct shapes. Our model assumed rectangular projections for computational simplicity and alignment with predetermined placements; future work could incorporate more detailed spatial partitioning to address this limitation. Third, more research is needed on the node evaluation function of IBS to improve its performance further. Fourth, considering different capacities of the PBF-LB/M machines is a crucial problem in practical applications and is worthy of further investigation. Finally, this study focused on a single 3D printing technology. Developing AMPS problems that incorporate multiple technologies would be useful for applications in 3D printing service bureaus/farms. Besides, newer

technologies, such as cold spray could benefit from tailored scheduling approaches, leveraging their competitive advantage of 'speed' over other metal AM processes.

CRediT authorship contribution statement

Pourya Pourhejazy: Writing – review & editing, Writing – original draft, Investigation. **Fei-Huan Lee:** Formal analysis, Data curation. **Kuo-Ching Ying:** Writing – review & editing, Supervision, Formal analysis, Conceptualization. **Shih-Wei Lin:** Validation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. . An Illustrative Example of the Computational Procedure of TIBS

Consider a small example with three parts, $i \in \{1, 2, 3\}$, to illustrate the computational steps of the TIBS algorithm. The heights, areas, and volumes of these parts are listed in Table A1. These parts will be processed on a 3D printer with a maximum build plate area of $a_m = 600$, a setup time of $s_t = 1.2$, a powder build-up time per layer of $h_t = 0.7$, and a molding time per unit volume of $v_t = 0.030864$. For this illustrative example, we will use an initial beam width of $D = 2$.

Table A1
Data for parts in the illustrative example.

Part (i)	Height (h_i)	Area (a_i)	Volume (v_i)	h_i^{nor}	v_i^{nor}
1	6.90	209.06	826.08	0.0000	0.8563
2	26.04	550.11	952.60	1.0000	1.0000
3	15.97	23.63	71.91	0.4738	0.0000

Step 1 of the algorithm involves generating an initial solution using the part data. In this case, the parts are sorted based on their height as $\{2, 3, 1\}$. A new batch is added if the total area of the parts exceeds 600 cubic centimeters. After placing parts 2 and 3 on the build plate, the available space is insufficient for the remaining parts, necessitating the addition of a new batch. The makespan of the resulting solution can be calculated as follows:

$$P_1 = s_t \cdot Z_1 + v_t \cdot \sum_{i \in I} v_i \cdot X_{1i} + h_t \cdot H_1 = 51.0485$$

$$P_2 = s_t \cdot Z_2 + v_t \cdot \sum_{i \in I} v_i \cdot X_{2i} + h_t \cdot H_2 = 31.5261$$

$C_{\max} = P_1 + P_2 = 82.5746$, and $j_n = 2$.

Step 2 determines the root node and sets the initial beam width to $D = 2$. The height and volume of the parts are then normalized as shown in Table A1.

In *Step 3*, the branching from the root node is completed, and part one is inserted into possible job batches to form the next layer of nodes. This branching process creates multiple potential solutions for further evaluation and exploration. The resulting structure of the nodes and their connections is shown in Figure A1.

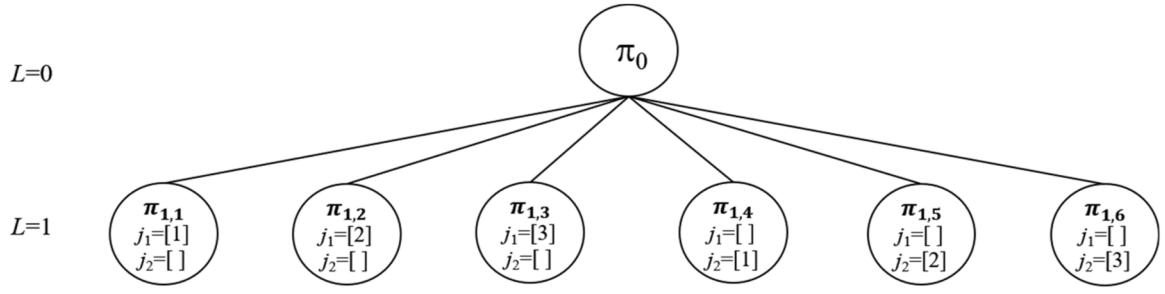

Fig. A1. Illustration of Step 3.

.

In *Step 4*, the feasibility of the schedules is first checked, and any equivalent nodes are deleted. After this process, there are a total of three remaining nodes: $\pi_{1,1}, \pi_{1,2}, \pi_{1,3}$. In *Step 5*, the lower bounds of the remaining nodes are calculated as follows:

$$Bound(\pi_{1,1}) = 1.2 + (0.030864 * 826.08) + 0.7 * 6.9 + 0.030864 * (952.6 + 71.91) \quad = 63.1466$$

$$Bound(\pi_{1,2}) = 1.2 + (0.030864 * 952.6) + 0.7 * 26.04 + 0.030864 * (826.08 + 71.91) = 76.5446$$

$$Bound(\pi_{1,3}) = 1.2 + (0.030864 * 71.91) + 0.7 * 15.97 + 0.030864 * (826.08 + 952.6) = 69.4956$$

By using the makespan of the initial solution as the UB, it is observed that the lower bounds of all nodes in the first layer are better (*i.e.*, lower) than the UB. Therefore, these nodes are identified as promising and are retained for further consideration in the search process.

Next, the fitness function value of each node is determined, and the $D = 2$ nodes with the best scores are reserved. The scores of the nodes are calculated as follows:

$$Score(\pi_{1,1}) = \frac{0.7}{0.7 + 0.030864} \times 0 + \left(1 - \frac{0.7}{0.7 + 0.030864}\right) \times 0.8563 = 0.03614$$

$$Score(\pi_{1,2}) = \frac{0.7}{0.7 + 0.030864} \times 1 + (1 - \frac{0.7}{0.7 + 0.030864}) \times 1 = 1$$

$$Score(\pi_{1,3}) = \frac{0.7}{0.7 + 0.030864} \times 0.4738 + (1 - \frac{0.7}{0.7 + 0.030864}) \times 0 = 0.4538.$$

Therefore, $\pi_{1,1}$ and $\pi_{1,3}$ are given further consideration in the subsequent computation rounds. The new rounds of branching and evaluation are illustrated in Figure A2.

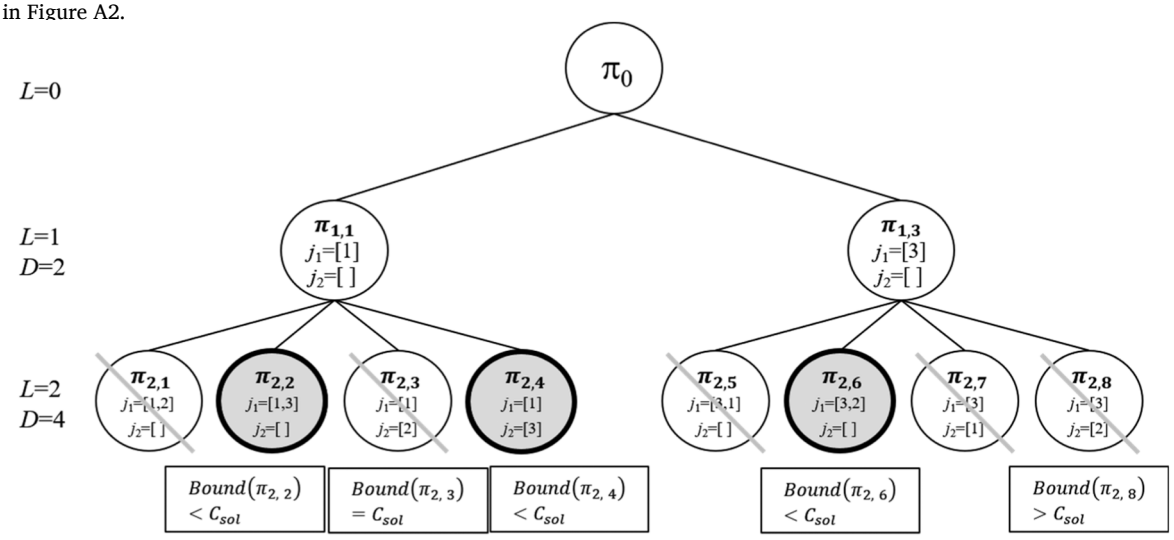

Fig. A2. Illustration of branching and evaluation in the second layer.

.

Branching and evaluation of nodes under $\pi_{1,3}$ on the left-hand side of the diagram reveal that $\pi_{2,7}$ (*i.e.*, node seven from the left-hand side in the second layer) violates the allowed production area; therefore, it should be removed. Considering the LB values and $D = 4$, the bound values are calculated as follows. Based on these values, the best four nodes must be reserved to further branching until all parts are assigned to production batches.

$$Bound(\pi_{2,2}) = 1.2 + 0.03086 \times (826.08 + 71.91) + 0.7 \times 15.97 + 0.03086 \times 952.6 = 69.4956$$

$$Bound(\pi_{2,3}) = 2.4 + 0.03086 \times (826.08 + 952.6) + 0.7 \times (6.9 + 26.04) + 0.03086 \times 71.91 = 82.5746$$

$$Bound(\pi_{2,4}) = 2.4 + 0.03086 \times (826.08 + 71.91) + 0.7 \times (6.9 + 15.97) + 0.03086 \times 952.6 = 75.5256$$

$$Bound(\pi_{2,6}) = 1.2 + 0.03086 \times (952.6 + 71.91) + 0.7 \times 26.04 + 0.03086 \times 826.08 = 76.5446$$

$$Bound(\pi_{2,8}) = 2.4 + 0.03086 \times (71.91 + 952.6) + 0.7 \times (15.97 + 26.04) + 0.03086 \times 826.08 = 88.92$$

In the illustrative example, due to the small number of parts, only three nodes remain after the screening procedure in the second layer ($\pi_{2,2}$, $\pi_{2,4}$, *and*$\pi_{2,6}$), as these nodes have lower bounds below the UB. To proceed with the unassigned parts, all three nodes are directly branched to the next level as there are fewer than $D = 4$. Figure A3 illustrates the branching and evaluation procedure for assigning the last part.

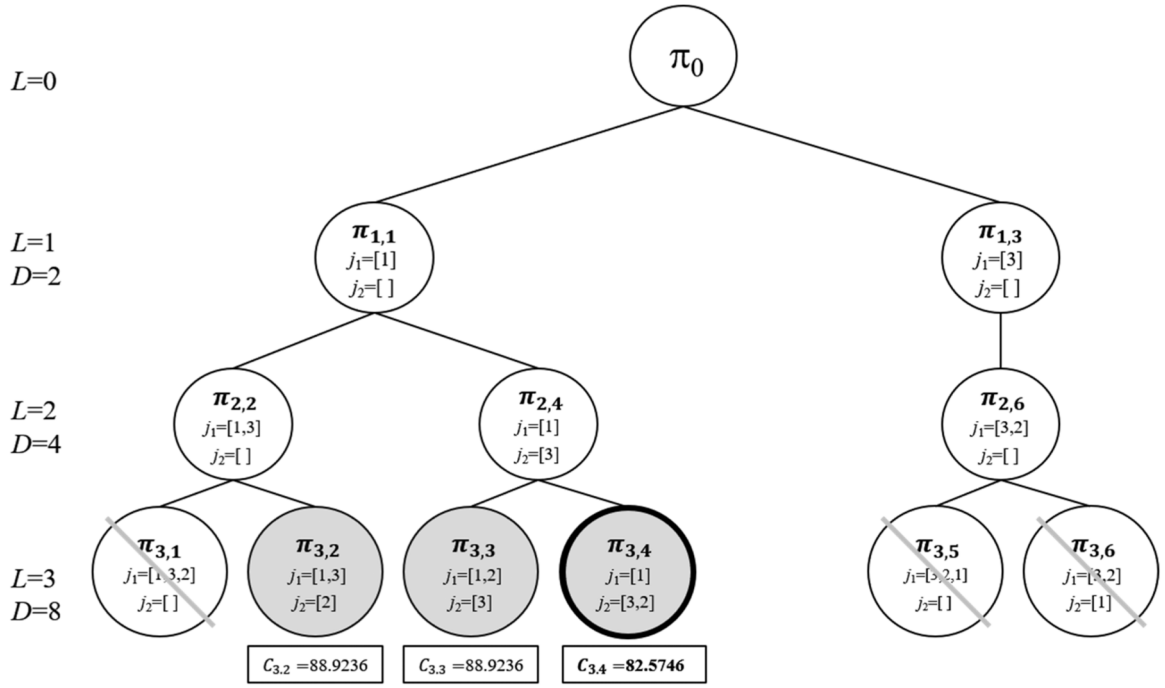

Fig. A3. Illustration of branching and evaluation in the third layer.

.

After removing infeasible and equivalent nodes, $\pi_{3,2}$, $\pi_{3,3}$, and $\pi_{3,4}$ are further evaluated to find the best solution; The calculation of the makespans for these solutions is as follows. The resulting production schedule has been selected from $\pi_{3,2}$, $\pi_{3,3}$, and $\pi_{3,4}$. Therefore, TIBS terminates by selecting $\pi_{3,4}$ as the final output.

$$C(\pi_{3,2}) = 2.4 + 0.03086 \times (826.08 + 71.91 + 952.6) + 0.7 \times (15.97 + 26.04) + 0.03086 \times 826.08 = 88.92$$

$$C(\pi_{3,3}) = 2.4 + 0.03086 \times (826.08 + 71.91 + 952.6) + 0.7 \times (15.97 + 26.04) + 0.03086 \times 826.08 = 88.92$$

$$C(\pi_{3,4}) = 2.4 + 0.03086 \times (826.08 + 71.91 + 952.6) + 0.7 \times (6.9 + 26.04) + 0.03086 \times 826.08 = 82.5746$$

References

[1] Pourhejazy P. Additive manufacturing in the supply chain. The Palgrave Handbook of Supply Chain Management. Cham: Springer International Publishing,; 2024. p. 1383–403. https://doi.org/10.1007/978-3-031-19884-7_110.

[2] Martínez-García A, Monzón M, Paz R. Standards for additive manufacturing technologies. Additive Manufacturing. Elsevier; 2021. p. 395–408. https://doi.org/10.1016/B978-0-12-818411-0.00013-6.

[3] Ying K-C, Fruggiero F, Pourhejazy P, Lee B-Y. Adjusted Iterated Greedy for the optimization of additive manufacturing scheduling problems. Expert Syst Appl 2022;198:116908. https://doi.org/10.1016/j.eswa.2022.116908.

[4] Kucukkoc I. MILP models to minimise makespan in additive manufacturing machine scheduling problems. Comput Oper Res 2019;105:58–67. https://doi.org/10.1016/j.cor.2019.01.006.

[5] Afrasiabi M, Bambach M. Modelling and simulation of metal additive manufacturing processes with particle methods: a review. Virtual Phys Prototyp 2023;18. https://doi.org/10.1080/17452759.2023.2274494.

[6] Liu Y, Sing SL. A review of advances in additive manufacturing and the integration of high-performance polymers, alloys, and their composites. Mater Sci Addit Manuf 2023;2:1587. https://doi.org/10.36922/msam.1587.

[7] Rao J, Leong Sing S, Liu P, Wang J, Sohn H. Non-destructive testing of metal-based additively manufactured parts and processes: a review. Virtual Phys Prototyp 2023;18. https://doi.org/10.1080/17452759.2023.2266658.

[8] Ying K-C, Pourhejazy P, Huang Y-H. Tailored Iterated Greedy metaheuristic for a scheduling problem in metal 3D printing. Adv Eng Softw 2023;186:103546. https://doi.org/10.1016/j.advengsoft.2023.103546.

[9] Dall'Agnol G, Sagawa JK, Tavares Neto RF. Scheduling for additive manufacturing: a literature review. Gest Produção 2022;29. https://doi.org/10.1590/1806-9649-2022v29e1922.

[10] Coruzzolo AM, Balugani E, Gamberini R. Spare parts management with additive manufacturing (AM): a critical review. IFAC-Pap 2022;55:1159–64. https://doi.org/10.1016/j.ifacol.2022.09.546.

[11] Framinan JM, Perez-Gonzalez P, Fernandez-Viagas V. An overview on the use of operations research in additive manufacturing. Ann Oper Res 2023;322:5–40. https://doi.org/10.1007/s10479-022-05040-4.

[12] Fera M, Fruggiero F, Lambiase A, Macchiaroli R, Todisco V. A modified genetic algorithm for time and cost optimization of an additive manufacturing single-machine scheduling. Int J Ind Eng Comput 2018:423–38. https://doi.org/10.5267/j.ijiec.2018.1.001.

[13] Fera M, Macchiaroli R, Fruggiero F, Lambiase A. A modified tabu search algorithm for the single-machine scheduling problem using additive manufacturing technology. Int J Ind Eng Comput 2020:401–14. https://doi.org/10.5267/j.ijiec.2020.1.001.

[14] Arık OA. Additive manufacturing scheduling problem considering assembly operations of parts. Oper Res 2021. https://doi.org/10.1007/s12351-021-00649-y.

[15] Alicastro M, Ferone D, Festa P, Fugaro S, Pastore T. A reinforcement learning iterated local search for makespan minimization in additive manufacturing machine scheduling problems. Comput Oper Res 2021;131:105272. https://doi.org/10.1016/J.COR.2021.105272.

[16] Wu Q, Xie N, Zheng S, Bernard A. Online order scheduling of multi 3D printing tasks based on the additive manufacturing cloud platform. J Manuf Syst 2022;63:23–34. https://doi.org/10.1016/j.jmsy.2022.02.007.

[17] Liu L, Wu Z, Yu Y. A branch-and-price algorithm to perform single-machine scheduling for additive manufacturing. J Manag Sci Eng 2023;8:273–86. https://doi.org/10.1016/j.jmse.2022.10.001.

[18] Che Y, Hu K, Zhang Z, Lim A. Machine scheduling with orientation selection and two-dimensional packing for additive manufacturing. Comput Oper Res 2021;130:105245. https://doi.org/10.1016/J.COR.2021.105245.

[19] Kapadia MS, Uzsoy R, Starly B, Warsing DP. A genetic algorithm for order acceptance and scheduling in additive manufacturing. Int J Prod Res 2021:1–18. https://doi.org/10.1080/00207543.2021.1991023.

[20] Aloui A, Hadj-Hamou K. A heuristic approach for a scheduling problem in additive manufacturing under technological constraints. Comput Ind Eng 2021;154:107115. https://doi.org/10.1016/j.cie.2021.107115.
[21] Rohaninejad M, Tavakkoli-Moghaddam R, Vahedi-Nouri B, Hanzálek Z, Shirazian S. A hybrid learning-based meta-heuristic algorithm for scheduling of an additive manufacturing system consisting of parallel SLM machines. Int J Prod Res 2021:1–21. https://doi.org/10.1080/00207543.2021.1987550.
[22] Altekin FT, Bukchin Y. A multi-objective optimization approach for exploring the cost and makespan trade-off in additive manufacturing. Eur J Oper Res 2022;301:235–53. https://doi.org/10.1016/j.ejor.2021.10.020.
[23] Karimi S, Kwon S, Ning F. Energy-aware production scheduling for additive manufacturing. J Clean Prod 2021;278:123183. https://doi.org/10.1016/J.JCLEPRO.2020.123183.
[24] De Antón J, Villafáñez F, Poza D, López-Paredes A. A framework for production planning in additive manufacturing. Int J Prod Res 2022:1–18. https://doi.org/10.1080/00207543.2022.2160026.
[25] He P, Li K, Kumar PNR. An enhanced branch-and-price algorithm for the integrated production and transportation scheduling problem. Int J Prod Res 2022;60:1874–89. https://doi.org/10.1080/00207543.2021.1876941.
[26] Zehetner D, Gansterer M. The collaborative batching problem in multi-site additive manufacturing. Int J Prod Econ 2022;248:108432. https://doi.org/10.1016/j.ijpe.2022.108432.
[27] Kim J, Kim H-J. An exact algorithm for an identical parallel additive machine scheduling problem with multiple processing alternatives. Int J Prod Res 2022;60:4070–89. https://doi.org/10.1080/00207543.2021.2007426.
[28] Kim J, Kim H-J. Parallel machine scheduling with multiple processing alternatives and sequence-dependent setup times. Int J Prod Res 2021;59:5438–53. https://doi.org/10.1080/00207543.2020.1781278.
[29] He J, Wu J, Zhang Y, Wang Y, He H. Large-scale customized production scheduling of multiagent-based medical 3d printing. Comput Intell Neurosci 2022;2022:1–13. https://doi.org/10.1155/2022/6557137.
[30] Oh Y, Cho Y. Scheduling of build and post processes for decomposed parts in additive manufacturing. Addit Manuf 2022:103164. https://doi.org/10.1016/j.addma.2022.103164.
[31] Hu K, Che Y, Zhang Z. Scheduling unrelated additive manufacturing machines with practical constraints. Comput Oper Res 2022;144:105847. https://doi.org/10.1016/j.cor.2022.105847.
[32] Toksarı MD, Toğa G. Single batch processing machine scheduling with sequence-dependent setup times and multi-material parts in additive manufacturing. CIRP J Manuf Sci Technol 2022;37:302–11. https://doi.org/10.1016/j.cirpj.2022.02.007.
[33] Zipfel B, Neufeld J, Buscher U. An iterated local search for customer order scheduling in additive manufacturing. Int J Prod Res 2023:1–21. https://doi.org/10.1080/00207543.2023.2167015.
[34] Poudel L, Elagandula S, Zhou W, Sha Z. Decentralized and centralized planning for multi-robot additive manufacturing. J Mech Des 2023;145. https://doi.org/10.1115/1.4055735.
[35] Nascimento P, Silva C, Mueller S, Moniz S. Nesting Sched Addit Manuf: Approach Considering Order Due Dates 2023:117–28. https://doi.org/10.1007/978-3-031-20788-4_8.
[36] Lee SJ, Kim BS. Two-stage meta-heuristic for part-packing and build-scheduling problem in parallel additive manufacturing. Appl Soft Comput 2023;136:110132. https://doi.org/10.1016/j.asoc.2023.110132.
[37] Ying K-C, Lin S-W. Minimizing makespan in two-stage assembly additive manufacturing: a reinforcement learning iterated greedy algorithm. Appl Soft Comput 2023;138:110190. https://doi.org/10.1016/j.asoc.2023.110190.
[38] Ow PS, Morton TE. Filtered beam search in scheduling. Int J Prod Res 1988;26:35–62. https://doi.org/10.1080/00207548808947840.
[39] Valente JMS, Alves RAFS. Filtered and recovering beam search algorithms for the early/tardy scheduling problem with no idle time. Comput Ind Eng 2005;48:363–75. https://doi.org/10.1016/J.CIE.2005.01.020.
[40] Pourhejazy P, Lin S-W, Cheng C-Y, Ying K-C, Lin P-Y. Improved beam search for optimizing no-wait flowshops with release times. IEEE Access 2020;8:148100–24. https://doi.org/10.1109/ACCESS.2020.3015737.
[41] Fernandez-Viagas V, Framinan JM. A beam-search-based constructive heuristic for the PFSP to minimise total flowtime. Comput Oper Res 2017;81:167–77. https://doi.org/10.1016/J.COR.2016.12.020.
[42] Ghirardi M, Potts CN. Makespan minimization for scheduling unrelated parallel machines: a recovering beam search approach. Eur J Oper Res 2005;165:457–67. https://doi.org/10.1016/J.EJOR.2004.04.015.
[43] Libralesso L, Focke PA, Secardin A, Jost V. Iterative beam search algorithms for the permutation flowshop. Eur J Oper Res 2022;301:217–34. https://doi.org/10.1016/J.EJOR.2021.10.015.
[44] Li Q, Zhang D, Wang S, Kucukkoc I. A dynamic order acceptance and scheduling approach for additive manufacturing on-demand production. Int J Adv Manuf Technol 2019;105:3711–29. https://doi.org/10.1007/s00170-019-03796-x.