



ARTICLE

Scheduling of Heterogeneous FDM Printer Farms under Sequence-Dependent Setups and Maintenance: A Discrete Random-Key GBFIO (DR-GBFIO) Approach

Samet DİRİ^{*}

Department of Information Systems Engineering, Kocaeli University, Kocaeli, Türkiye

*Corresponding Author: Samet DİRİ. Email: samet.diri@kocaeli.edu.tr

Received: 20 August 2026; Accepted: 23 September 2026; Published: 21 October 2026

ABSTRACT: The scheduling of heterogeneous parallel machines under complex operational constraints is a computationally demanding NP-hard combinatorial optimization problem. While often encountered in modern production farms, resolving sequence-dependent setups and mandatory maintenance windows requires advanced metaheuristic frameworks. This study formulates an integrated scheduling model for joint job-to-machine assignment and sequencing, utilizing a heterogeneous Fused Deposition Modeling (FDM) printer farm as the application domain; the model requires every job to be assigned and completed, and its primary objective is the minimization of the maximum completion time (makespan), with an extended weighted-sum multi-objective formulation also provided. To solve this problem, the recently proposed Grizzly Bear Fat Increase Optimizer (GBFIO) is faithfully adapted to the discrete domain via random-key encoding and an event-driven decoder, yielding the proposed Discrete Random-Key GBFIO (DR-GBFIO). DR-GBFIO is computationally benchmarked against Particle Swarm Optimization (PSO), a Genetic Algorithm (GA), and Harris Hawks Optimization (HHO) across three problem scales reaching up to 20 machines and 100 jobs over 1200 optimization runs, evaluated via Friedman and Wilcoxon signed-rank tests. Results demonstrate that DR-GBFIO significantly outperforms all baselines across all three problem scales, with an approximate 4% to 18% reduction in makespan, a pattern consistent with preserving the original equations, including multi-source attraction, linearly decaying envelopes, and repeated intensification, without approximation. This study contributes a robust metaheuristic framework and one of the first statistically validated, equation-faithful adaptations of GBFIO to a combinatorial context.

KEYWORDS: Metaheuristic optimization; combinatorial scheduling; discrete random-key encoding; sequence-dependent setup times; heterogeneous parallel machines; DR-GBFIO

1 Introduction

Fused Deposition Modeling (FDM)-based 3D printer farms represent a production model that is becoming increasingly widespread in prototyping, spare-part production, and small-to-medium batch manufacturing. A farm typically comprises heterogeneous printers of different brands and models, differing in build volume, print speed, and material support. Unlike classical identical parallel-machine scheduling problems, this heterogeneity jointly foregrounds three elements:

- machine-dependent processing times,
- sequence-dependent setup times arising from material and color changes,
- a geometric feasibility constraint that substitutes for a capacity constraint.

In addition, the periodic preventive-maintenance (PM) requirement of the printers places the problem within the class of heterogeneous parallel-machine scheduling problems with maintenance windows and sequence-dependent setup times.

The scheduling problem addressed in this study generalizes the classical unrelated parallel-machine scheduling problem with sequence-dependent setup times, commonly denoted $R|s_{jk}|C_{max}$ in the standard three-field scheduling notation. Since this problem in turn generalizes the identical parallel-machine makespan-minimization problem $P||C_{max}$, which is known to be NP-hard through a straightforward reduction from the PARTITION problem, and the addition of machine heterogeneity, sequence-dependent setup times, and mandatory maintenance windows can only add further combinatorial complexity, the problem studied here is NP-hard as well, consistent with both the classical complexity-theoretic treatment of such reductions [1] and scheduling-specific treatments of parallel-machine complexity results [2]. Because this problem class is NP-hard, exact solution methods fail to reach optimal solutions within a reasonable time for realistically sized instances, which necessitates metaheuristic approaches. Although methods such as iterated greedy, tabu search, genetic algorithms, and bee colony optimization have been widely used in the additive-manufacturing scheduling literature, the adaptation of newer nature-inspired algorithms to combinatorial scheduling problems remains limited. This study addresses this gap through the Grizzly Bear Fat Increase Optimizer (GBFIO) [3], a continuous optimization algorithm inspired by grizzly bear foraging behavior.

In the model formulated in this study, every job must eventually be assigned to a printer and completed; no job may be left unscheduled or rejected. Among the possible objectives for such a schedule, this study adopts the minimization of the maximum completion time (makespan) as its primary objective, since it is the performance measure most directly comparable across the scheduling literature; an extended, weighted-sum multi-objective formulation that additionally accounts for total setup time and printer utilization is also provided, though it is analyzed conceptually rather than benchmarked experimentally.

The objectives of this study are:

- to develop a mathematical scheduling model that jointly addresses material changeover, color change, build-volume/material feasibility, and preventive maintenance for a heterogeneous FDM printer farm;
- to adapt GBFIO to the discrete scheduling problem via random-key encoding;
- to benchmark the proposed approach against classical and recent metaheuristics within a multi-scale, statistically validated experimental framework.

The novelty of this study is twofold. First, while sequence-dependent setup times and preventive maintenance have each been modeled separately for parallel-machine scheduling, very few studies integrate machine heterogeneity, dual material-and-color setup penalties, geometric/material feasibility, and mandatory maintenance windows within a single combinatorial model. This study closes that gap through an explicit formulation together with an always-feasible random-key encoding that guarantees valid schedules without penalty functions. Second, GBFIO has not previously been adapted to a discrete combinatorial domain. This study provides one of the first equation-faithful adaptations, preserving the algorithm's original search architecture rather than approximating it heuristically. Validated through 1200 optimization runs and non-parametric statistical testing, the study offers the scheduling community a reusable problem formulation together with methodological evidence on how equation-fidelity affects an adapted algorithm's performance, a question of broader interest to researchers adapting new-generation metaheuristics to combinatorial engineering problems.

The remainder of the paper is organized as follows: [Section 2](#) reviews the related literature and clarifies the research gap; [Section 3](#) presents the problem definition; [Section 4](#) formulates the mathematical

model; [Section 5](#) describes the proposed discrete GBFIO approach; [Section 6](#) defines the experimental design; [Section 7](#) presents the findings together with the statistical tests; and [Section 8](#) concludes the paper.

2 Related Work

2.1 Scheduling and Nesting Dynamics in Additive Manufacturing

The most salient feature of additive manufacturing scheduling is that the assignment of jobs to machines (scheduling) is intertwined with their placement on two- or three-dimensional build platforms (nesting). Li et al. [4] addressed the problem of order planning in additive manufacturing systems, proposing a Mixed-Integer Linear Programming (MILP) model and heuristic methods for the joint optimization of machine selection and production cost, thereby demonstrating the critical role of this integration in reducing machine utilization time and overall production cost. Chergui et al. [5] advanced this structure further by addressing batch formation (1D bin-packing) jointly with scheduling decisions through a two-stage MILP and heuristic approach, focusing on tardiness minimization and demonstrating the importance of jointly optimizing geometric placement with scheduling decisions. Kucukkoc [6] developed separate MILP models for single-machine, identical parallel-machine, and unrelated parallel-machine environments, mathematically establishing the reducing effect of batch formation dynamics on makespan. Kim and Kim [7] addressed a problem class directly related to this study by proposing a part-grouping and scheduling problem with sequence-dependent setup times for non-identical parallel additive manufacturing machines. This study is notable as one of the rare examples in the literature that jointly considers heterogeneous machines, sequence-dependent setup times, and makespan minimization; however, it treats material change as the sole source of setup time and does not incorporate color change or preventive maintenance. Oh et al. [8] systematized nesting and scheduling problems through a comprehensive taxonomy and review, showing that the field has evolved since the mid-2010s toward studies that jointly address nesting and scheduling. A recent bibliometric study updating this mapping [9] shows that heuristic and metaheuristic methods dominate the nesting-scheduling literature, but that the full integration of placement and scheduling remains limited.

2.2 Multi-Objective Optimization and Sustainability

As in conventional manufacturing, minimizing makespan alone is not sufficient in additive manufacturing. Ransikarbum et al. [10] addressed the part-to-printer assignment problem in heterogeneous FDM printer farms within a multi-objective framework, presenting analytical models that simultaneously optimize production cost, load balancing, and total time. The concept of green scheduling has become prominent in recent literature. Because the heating (bed and nozzle) and cooling processes of additive manufacturing machines consume substantial energy, reducing machine idle time and allocating jobs according to energy-consumption rates have become a critical optimization component for sustainability; the green unrelated parallel-machine scheduling model proposed by [11], which also incorporates preventive maintenance, is a recent example of this sustainability-oriented approach and constitutes a direct reference point for the energy indicator component of the proposed model.

2.3 Metaheuristic Solution Approaches and Constraint Management

Problem-solving approaches for combinatorial optimization problems such as scheduling are commonly organized into four broad classes. Exact (conventional) methods, such as branch-and-bound and mixed-integer linear programming, guarantee optimal solutions but scale poorly with problem size. Problem-specific heuristics, such as greedy construction rules or dispatching rules, produce fast but not necessarily high-quality solutions using domain knowledge tailored to a single problem. A metaheuristic is a general-purpose, problem-independent search strategy that guides an underlying heuristic (typically

through iterative population-based or trajectory-based exploration and exploitation mechanisms) toward high-quality, though not guaranteed optimal, solutions across a broad range of problem types; genetic algorithms, particle swarm optimization, simulated annealing, and tabu search are long-established examples, while GBFIO and Harris Hawks Optimization represent more recent, nature-inspired instances of this class. Artificial intelligence and machine learning methods, such as reinforcement learning or neural-network-based prediction, instead learn solution policies or performance estimates from data and are frequently combined with the first three classes in hybrid frameworks, as seen in the Q-learning-based metaheuristic discussed below. This study is positioned in the third class, adapting a nature-inspired metaheuristic to a discrete scheduling problem.

The heterogeneous nature of printers in production environments, together with sequence-dependent setup times and maintenance constraints, inherently places the problem in the NP-hard class. Because classical integer linear programming models can only yield optimal results for small-scale problems, researchers have turned to nature-inspired metaheuristic algorithms. Ref. [12] proposed a Tailored Iterated Greedy algorithm (TIG) that accounts for batch formation and limited build-platform constraints in metal 3D printing, demonstrating the effectiveness of this metaheuristic on industrial-scale problems; however, this study operates under an identical-machine assumption and does not address heterogeneous printer fleets or color change. Among classical metaheuristics, Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) are frequently used as baselines in these problems, although hybrid or new-generation exploitation-oriented algorithms are increasingly preferred today. On the other hand, the continuous operation of printer farms necessitates periodic maintenance to prevent machine failures. The model developed by [13] for unrelated parallel-machine environments, focusing on dependent setup times and preventive maintenance, constitutes a fundamental reference for formulating maintenance windows as constraints in contemporary additive-manufacturing scheduling problems and forms the conceptual basis of this study's maintenance-obligation formulation; unlike the present study, however, it does not incorporate the additive-manufacturing context, printer heterogeneity, or color-change elements. A comprehensive survey on the topic [14] likewise emphasizes that studies jointly addressing sequence-dependent setup times and maintenance constraints remain relatively scarce in the literature. A systematic review of multi-objective scheduling metaheuristics in the Industry 4.0/5.0 context [15] shows that research intensity in intelligent manufacturing scheduling has increased markedly since 2020, although studies incorporating three or more objective functions remain relatively rare.

On the algorithmic side, GBFIO [3] is a nature-inspired continuous optimization algorithm that models grizzly bear feeding behaviors (food search, hunting, fishing) through a three-phase search mechanism. The original study evaluated the algorithm's performance on standard benchmark functions, the CEC2017 test suite, and four constrained engineering design problems. However, the literature review conducted for this study found that the algorithm's adaptation to combinatorial/scheduling problems has not yet been tested.

Within computer-modeling venues, metaheuristic scheduling for heterogeneous production resources has likewise received sustained attention: Ref. [16] formulated an energy-aware parallel-machine scheduling model under deteriorating and learning effects with stochastic processing times, jointly minimizing makespan and energy consumption; Ref. [17] proposed a two-population collaborative evolutionary algorithm for multi-objective flexible job-shop scheduling; and Ref. [18] addressed a dual-resource-constrained distributed flexible job-shop problem using a Q-learning-based multi-objective Grey Wolf Optimizer (GWO), jointly minimizing makespan and energy consumption. This body of work confirms that energy- and constraint-aware heterogeneous-machine scheduling, solved via metaheuristics that preserve their original nature-inspired formulations, is an active and recognized research direction.

The adaptation of new-generation metaheuristics such as GBFIO and HHO [19]—which possess a nonlinear exploration-exploitation balance capable of overcoming the premature-convergence problems of classical algorithms such as GA or PSO in high-dimensional search spaces—to the discrete scheduling domain therefore represents a potential new avenue of research in the literature.

2.4 Contribution of This Study

Table 1 presents a comparison of the related studies with respect to the five core elements addressed in this research (printer/machine heterogeneity, material change, color change, capacity/build-volume constraint, preventive maintenance), together with the energy indicator, metaheuristic solution approach, and statistical validation.

Table 1: Comparison of related studies with respect to the model elements addressed.

Study	Heterogeneous Machine	Material Change	Color Change	Capacity/Build Volume	Preventive Maintenance	Energy	Metaheuristic	Statistical Validation
[4]	✗ (identical)	✗	✗	✓ (1D)	✗	✗	✗ (MILP+heuristic)	Partial
[5]	✗ (identical)	✗	✗	✓ (1D)	✗	✗	✗ (MILP+heuristic)	Partial
[13]	✓	✓ (general)	✗	✗	✓	✗	✓	Partial
[6]	✓ (some models)	✗	✗	✓ (1D)	✗	✗	✗ (MILP)	Partial
[10]	✓	✗	✗	✗	✗	✗	✗ (analytical)	Partial
[7]	✓	✓ (dependent)	✗	✓ (2D area)	✗	✗	✓ (metaheuristic)	✓
[12]	✗ (identical)	✗ (independent)	✗	✓	✗	✗	✓ (TIG)	✓
[11]	✓	✗	✗	✗	✓	✓	✓ (math.)	Partial
This study	✓	✓ (dependent)	✓ (dependent)	✓ (3D)	✓	✓	✓ (4 algorithms)	✓

Table 1 shows that the proposed study is among the limited number of studies in literature that jointly address all these elements. The number of studies that jointly model material change, color change, and preventive maintenance in the context of a heterogeneous FDM printer farm is limited in the literature. Most existing studies address only one or two of these elements simultaneously. For example, Ref. [7] combines heterogeneity and sequence-dependent setup time but does not incorporate maintenance. Ref. [11] combines maintenance and energy but does not address setup times. Color change is rarely modeled in literature as a setup-time source independent of material change; none of the reviewed studies explicitly address color change. The full integration of nesting/placement with scheduling remains limited [8,9]. While this study treats build-volume feasibility as a constraint, it excludes full 3D geometric placement (nesting) from its scope; this is stated as future work in a later section. The adaptation of new-generation nature-inspired algorithms to combinatorial scheduling problems remains limited in the literature. Considering the findings of our literature review, this adaptation has not previously been reported in the existing literature for GBFIO specifically.

3 Problem Definition and Assumptions

The system under consideration consists of a printer farm $P = \{P_1, \dots, P_m\}$ comprising m heterogeneous FDM printers and a job set $J = \{J_1, \dots, J_n\}$ comprising n print jobs. The optimization problem corresponds to two nested decisions:

- which printer each job is assigned to,
- the processing sequence on the assigned printer.

3.1 Printer Properties

Each machine $i \in P$ in the heterogeneous printer farm considered in this study is defined by five core technical properties that directly affect scheduling decisions and system constraints.

3.1.1 Build Volume (B_i)

Expresses the maximum print volume the printer can provide, in millimeters ($X_i \times Y_i \times Z_i$). For a job to be assigned to the corresponding printer, the part dimensions must remain within these volume limits; this operates in the model as a geometric feasibility constraint.

3.1.2 Print Speed Coefficient (v_i)

A proportional multiplier that affects job production times based on the machine's kinematic structure and extrusion capacity. The actual processing time of job j on printer i (p_{ij}) is obtained by dividing the job's reference time by this coefficient.

3.1.3 Supported Material Set (M_i)

The subset of filament types (e.g., PLA, ABS, PETG) that the printer's hardware configuration (nozzle temperature, heated-bed requirement, etc.) is capable of processing. Together with build volume, it is one of the fundamental hard constraints determining which jobs can be assigned to which printers.

3.1.4 Preventive Maintenance Parameters (I_i and D_i^{maint})

Specifies the periodic maintenance interval, defined as a function of cumulative operating time to sustain the machine's operational reliability (I_i , hours), and the mandatory downtime required by this procedure (D_i^{maint} , minutes).

3.1.5 Energy Consumption Rate (e_i)

Represents the average power (kW) drawn from the grid per unit time during active printing. This parameter is included in the model to measure the environmental sustainability of the production farm and to compute energy-oriented secondary performance indicators.

3.2 Job Properties

Each print job $j \in J$ awaiting processing in the system is defined by three core properties that directly affect assignment and sequencing decisions.

3.2.1 Reference Print Time (p_j)

Represents the base time (minutes) required to produce the job at a standard extrusion speed. This value is divided by the speed coefficient of the assigned printer (v_i) to determine the job's actual machine-specific processing time (p_{ij}).

3.2.2 Required Material (mat_j) and Color (col_j)

Defines the filament type and color requested for the print. While the material information serves as a filter in the feasibility matrix that determines which printers a job may be assigned to, both material and color information are fundamental in determining the sequence-dependent total setup times between consecutive jobs (e.g., the transition from j to k), given by (S_{jk}).

3.2.3 Part Dimensions (b_j)

Specifies the three-dimensional outer dimensions of the part to be produced, in millimeters ($x_j \times y_j \times z_j$). It is a structural hard constraint that, for a job to be assigned to a specific printer, these dimensions must not exceed the printer's build volume (B_i) limits along any axis.

3.2.4 Model Assumptions

In addition to the parametric definitions specified above, the proposed scheduling model is built on the following core assumptions:

- Each printer can process at most one job at a time, and once a print job begins, it cannot be interrupted under any circumstances (non-preemptive).
- Setup times (material/color change) are entirely sequence-dependent and are computed only by considering consecutive job pairs assigned to the same machine.
- Preventive maintenance activities are enforced strictly, without any flexibility margin. When a machine's cumulative operating time exceeds its maintenance interval, a mandatory maintenance block is inserted before the next job can start and cannot be skipped.
- At the start of the planning horizon ($t = 0$), all printers are idle and available for assignment.

These assumptions aim to represent industrial realism at a reasonable level of complexity while preserving the analytical tractability of the model.

4 Mathematical Model

4.1 Sets and Parameters

The sets and parameters constructed to mathematically express the heterogeneous machine properties, sequence-dependent setup times, preventive-maintenance windows, and job requirements inherent to the problem are presented in [Table 2](#).

Table 2: Sets and parameters constructed in this study.

Symbol	Description
$J = \{1, \dots, n\}$	Job set
$P = \{1, \dots, m\}$	Printer set
p_{ij}	Processing time of job j on printer $i = p_j/v_i$
s_{jk}^{mat}	Material-change time from j to k (15 min if different, 0 if same)
s_{jk}^{col}	Color-change time from j to k (c_{purge} min if different, 0 if same)
$S_{jk} = s_{jk}^{mat} + s_{jk}^{col}$	Total (composite) setup-time parameter incurred during the sequential transition from j to k
I_i, D_i^{maint}	Maintenance interval (hours) and maintenance duration (min) of printer i

(Continued)

Table 2 (continued)

Symbol	Description
e_i	Energy consumption rate of printer i (kW)
$B_i = (X_i, Y_i, Z_i)$	Build volume of printer i
$b_j = (x_j, y_j, z_j)$	Part dimensions of job j
M_i	Set of materials supported by printer i
BigM	A sufficiently large constant (the big-M coefficient used in disjunctive constraints; distinct from M_i)

4.2 Decision Variables

The assignment, sequencing, and timing decisions underlying the proposed mathematical model are formulated through the following decision variables:

$x_{ij} \in \{0, 1\}$: binary assignment variable equal to 1 if job j is assigned to printer i , and 0 otherwise.

$y_{ijk} \in \{0, 1\}$: binary sequencing variable equal to 1 if, on printer i , job j is processed immediately (consecutively) before job k , and 0 otherwise.

$C_j \geq 0$: continuous completion-time variable denoting the end of processing (and any associated setup) for job j on its assigned printer.

$MT_{it} \geq 0$: continuous variable denoting, on printer i , the planned start time of the t -th mandatory preventive-maintenance activity that must be performed.

To avoid unnecessary analytical complexity and over-specification in the model, the sequential setup times between jobs are incorporated into Constraint 1 as a direct function of the sequencing variables (y_{ijk}) and parameters (S_{jk}), rather than as an independent decision variable. Accordingly, the cumulative setup time on each printer i (TS_i) is not a variable sought by the model, but a reporting metric computed by Eq. (1).

$$TS_i = \sum_{j \in J} \sum_{k \in J} S_{jk} \cdot y_{ijk} \quad (1)$$

The *total setup time* indicator analyzed in the experimental findings section of this study is based on this analytical definition.

4.3 Constraints

The mathematical and operational constraints required for the proposed scheduling model to produce valid and feasible solutions are defined as follows.

Constraint 1—Single Processing and Assignment Integrity: Ensures that each job is assigned to exactly one printer, as formulated in Eq. (2):

$$\sum_{i \in P} x_{ij} = 1, \quad \forall j \in J \quad (2)$$

Non-overlap of jobs assigned to the same printer in the schedule is ensured through disjunctive constraints, given in Eq. (3):

$$C_k \geq C_j + p_{ik} + S_{jk} - \text{BigM} \cdot (1 - y_{ijk}), \quad \forall i, j, k \quad (3)$$

Here, $S_{jk} = s_{jk}^{mat} + s_{jk}^{col}$ represents the composite setup time incurred only during the sequential transition from j to k . The sequencing variables are linked to the assignment variables through the constraints defined in Eq. (4), ensuring that a precedence relation on printer i can only be active between two jobs both assigned to that printer:

$$y_{ijk} \leq x_{ij} \quad \text{and} \quad y_{ijk} \leq x_{ik}, \quad \forall i, j, k \quad (4)$$

In addition, each job has at most one immediate successor and at most one immediate predecessor. This is enforced by Eqs. (5) and (6), so that the jobs assigned to a given printer form a single ordered sequence rather than a branching structure:

$$\sum_{i \in P} \sum_{k \in J} y_{ijk} \leq 1, \quad \forall j \in J \quad (5)$$

$$\sum_{i \in P} \sum_{j \in J} y_{ijk} \leq 1, \quad \forall k \in J \quad (6)$$

Constraint 2—Build-Volume and Material Feasibility: For a job to be assigned to a printer, the part dimensions must fit within the build volume and the job's material must be supported by the printer, as required in Eq. (7):

$$x_{ij} = 1 \implies (x_j \leq X_i \wedge y_j \leq Y_i \wedge z_j \leq Z_i) \wedge mat_j \in M_i \quad (7)$$

To narrow the solution space and reduce model size, this geometric and hardware feasibility condition is filtered through a pre-computed binary feasibility matrix $F[i, j]$, and for infeasible pairs (i, j) , $x_{ij} = 0$ is fixed.

Constraint 3—Material-Change Time: Defines the setup time arising when consecutive jobs differ in material, computed as given in Eq. (8):

$$s_{jk}^{mat} = \begin{cases} 15, & mat_j \neq mat_k \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

Constraint 4—Color-Change Time: Defines the time required for nozzle cleaning when consecutive jobs differ in color, computed as given in Eq. (9):

$$s_{jk}^{col} = \begin{cases} c_{purge}, & col_j \neq col_k \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

In this study, $c_{purge} = 8$ min is used, representing a purge-time parameter that can be calibrated using real field data.

Constraints 5 and 6—Preventive Maintenance: Expresses the printers' periodic maintenance obligation. Let q index a job's position in the sequence assigned to printer i by the y_{ijk} variables, let j denote the job at position q with immediate predecessor k (or no predecessor if $q = 1$), and let $CT_{i,q}$ denote printer i 's cumulative operating time since its most recent maintenance activity, measured immediately before job j begins, with $CT_{i,1} = 0$. If $CT_{i,q} + S_{kj} + p_{ij}$ would exceed the maintenance interval I_i (converted to minutes),

a mandatory maintenance block of duration D_i^{maint} is inserted immediately before job j . The completion time C_j is then computed as given in Eq. (10):

$$C_j = \begin{cases} C_k + S_{kj} + D_i^{maint} + p_{ij}, & CT_{i,q} + S_{kj} + p_{ij} > 60, I_i \\ C_k + S_{kj} + p_{ij}, & \text{otherwise} \end{cases} \quad (10)$$

Following the processing of job j , the cumulative-time counter $CT_{i,q+1}$ for the next position in the sequence is updated as defined in Eq. (11):

$$CT_{i,q+1} = \begin{cases} p_{ij}, & CT_{i,q} + S_{kj} + p_{ij} > 60, I_i \\ CT_{i,q} + S_{kj} + p_{ij}, & \text{otherwise} \end{cases} \quad (11)$$

This maintenance block cannot be skipped under any circumstances. The variable MT_{it} introduced earlier records the resulting start time of the t -th maintenance activity performed on printer i , whenever the condition above is triggered.

4.4 Objective Functions

Efficiency in additive manufacturing operations should be measured not only by completing orders in the shortest possible time, but also by minimizing the time losses (and consequent extrusion waste) arising from material and color changes, and by balancing the workload across the machine fleet. To capture this multi-dimensional nature of the problem, the proposed mathematical model offers two distinct formulations designed to provide flexibility for both researchers and industrial practitioners: a single-objective baseline structure that enables fair and verifiable benchmarking of algorithm performance against literature standards, and an extended multi-objective structure in weighted-sum form that allows decision-makers to balance time, cost, and efficiency according to their production priorities.

4.4.1 Primary Objective Function

The core experimental design and algorithm comparisons in this study focus on minimizing the maximum completion time (*makespan*), defined in Eq. (12), which is the most widely accepted performance metric in the scheduling literature. Denoting the moment the last job leaves production, this metric determines the total busy time of the printer farm and therefore directly reflects the delivery speed of customer orders and the overall throughput capacity of the system.

$$\min F_1 = C_{\max} = \max_{j \in J} C_j \quad (12)$$

4.4.2 Extended Multi-Objective Model

In an industrial 3D printing farm, it is not sufficient merely to finish production quickly; reducing filament waste and time losses arising from material and color changes (total setup time), and balancing the workload across the machine fleet (average utilization rate), are also critical to operational efficiency. The formulation that gives decision-makers the flexibility to make trade-offs among these objectives according to their production strategies, and that optimizes the problem using a weighted-sum approach, is presented in Eq. (13):

$$\min F = w_1 \cdot C_{\max} + w_2 \cdot TS - w_3 \cdot U_{avg} \quad (13)$$

Here, $TS = \sum_{i \in P} TS_i$ denotes the total sequential setup time in the system, and U_{avg} denotes the average printer utilization rate. The utilization rate U_i for each printer i is computed as the fraction of its own operating span occupied by active printing, given by $U_i = \frac{\sum_{j \in J} p_{ij} x_{ij}}{\max_{j: x_{ij}=1} C_j}$, and averaged as $U_{avg} = \frac{1}{m} \sum_{i \in P} U_i$ over the printer farm. The parameters w_1, w_2 , and w_3 are weight coefficients reflecting the decision-maker's preferences. Total energy consumption, reported as a secondary indicator, is computed as $E = \sum_{i \in P} e_i \left(\frac{1}{60} \sum_{j \in J} p_{ij} x_{ij} \right)$ (converting processing minutes to hours to match e_i in kW), reflecting power drawn during active printing only; setup and maintenance operations are assumed not to draw production-level power and are therefore excluded from E .

The experimental comparisons in this study are conducted using the standard performance metric most directly comparable with the literature, *makespan* ($w_1 = 1, w_2 = 0, w_3 = 0$). Total setup time, average printer utilization rate, and total energy consumption are examined and reported in detail as secondary performance indicators.

5 Proposed Method: Discrete Random-Key GBFIO

5.1 Solution Encoding

To apply GBFIO's continuous-space update mechanism to the discrete problem without structurally altering it, scheduling solutions are encoded as an n -dimensional real-valued *random-key* vector $X \in [0, 1]^n$. This proposed adaptation is named Discrete Random-Key GBFIO (DR-GBFIO). The decode stage, which converts this continuous vector into a valid schedule, consists of three core steps.

5.1.1 Assignment

Machine selection is performed at this stage. Rather than randomly assigning each job j , only the pool of printers satisfying the pre-computed binary feasibility matrix (i.e., that the part dimensions fit within the build volume and the filament type is supported) is considered. The assignment is made by scaling the corresponding X_j component of the vector according to the size of this feasible-printer pool. This approach functions as an inherently valid (*always-feasible* or *repair-by-construction*) encoding mechanism that completely prevents infeasible individuals violating capacity and compatibility constraints from arising in the population.

5.1.2 Sequencing

Job sequencing is performed at this stage. The processing order among jobs assigned to the same printer is sorted using a secondary key obtained by a multiplicative scrambling of the corresponding X_j component. Specifically, the secondary key SK_j is computed as given in Eq. (14):

$$SK_j = (997 \cdot X_j) \bmod 1 \quad (14)$$

This produces a value that is effectively decorrelated from the printer-assignment scaling described above despite being derived from the same underlying random-key component, so that assignment and sequencing decisions are not artificially coupled. This procedure is a deterministic derivative of the widely used (*Smallest Position Value*—SPV) rule in the literature.

5.1.3 Event-Driven Scheduling and Constraint Management

An event-driven scheduler algorithm processes each printer's job queue from start to finish to construct the timeline. During this stage, setup times arising from material and color changes between consecutive

jobs, together with mandatory preventive-maintenance blocks that are inserted whenever the cumulative operating-time limit is reached, are dynamically added to the schedule, and the final completion time of each job is computed.

5.2 Adaptation of GBFIO's Three Phases to the Discrete Problem

In transferring metaheuristics designed for continuous space to permutation/assignment-based combinatorial problems, random-key encoding is a well-established method in the literature that allows an algorithm's original update equations to be used without modification [20]. In this study, the equations belonging to the three phases of the original GBFIO publication [3] (orig. Eqs. (1)–(21)) are applied directly, in place of continuous engineering design variables, to each component of the n -dimensional random-key vector, *elementwise*. That is, the algorithm's control architecture (coefficient definitions, linearly decaying envelopes, multi-source attraction structure) is preserved identically to the original formulation; only the search space is relocated from the engineering bounds $[X_{min}, X_{max}]$ to the random-key interval $[0, 1)$. Because the update equations can occasionally displace a component outside this interval, each component of the resulting vector is clipped to $[0, 0.999999]$ after every phase, matching the boundary-handling role that the engineering bounds $[X_{min}, X_{max}]$ play in the original formulation.

5.2.1 Phase 1—Food Search

This corresponds to the original Eqs. (1)–(11). In the original formulation, the bear moves toward attraction points representing, in order, the five best individuals in the population as fish, honey, shellfish, carrion, and plant sources, with decreasing weights reflecting decreasing nutritional value ($r_1^s > r_2^s > r_3^s > r_4^s > r_5^s$). The same structure is preserved in this study: the five best individuals in the current population are selected as $x_{fish}, \dots, x_{plants}$, and the position update formulated in Eq. (15) is applied with the original weight distributions ($[0, 2], [0, 1], [0, 0.5], [0, 0.25], [0, 0.125]$), from which the coefficients are sampled according to Eq. (16).

$$X_{new} = X + r_1(x_{fish} - X) + r_2(x_{honey} - X) + r_3(x_{shell} - X) + r_4(x_{corpse} - X) + r_5(x_{plants} - X) \quad (15)$$

$$r_i = c_i \cdot rand \cdot \sqrt{rand} \quad (16)$$

5.2.2 Phase 2—Hunting and Cub Care

This corresponds to the original Eqs. (12)–(18). In the hunting sub-step, the individual is drawn toward the current best solution ('prey') by a step scaled by the coefficient given in Eq. (17):

$$A = \alpha \cdot (2r_3^h - 1) \quad (17)$$

α decreases linearly, as in the original, from 2.5 to 0, thereby providing wide-step exploration in early iterations and small-step exploitation in later iterations. In the parallel 'care' sub-step, among three randomly selected individuals from the population, the best plays the role of 'coyote' and the other two play 'cubs', and a cub-loss exploitation move is produced using the coefficient given in Eq. (18):

$$B = \rho \cdot (2r_8^h - 1) \quad (18)$$

ρ decreases linearly from 1.25 to 0. The original Bernoulli selection rule ($\beta \leq 0.7$: the hunting result is selected if this condition holds, otherwise the care result is selected) is preserved exactly.

5.2.3 Phase 3—Fishing

This corresponds to the original Eqs. (19)–(21). In the original formulation, the bear locally improves the solution through repeated fishing attempts in a fixed area as formulated in Eqs. (19) and (20):

$$X_{new} = (1 + F) \cdot X \quad (19)$$

$$F = \eta \cdot \gamma \cdot \cos(2\pi r_1^f) \quad (20)$$

Here, $\eta = 0.3$ is a constant and γ is a linearly decaying envelope coefficient from 1 to 0. The original study repeats this step 25 times per individual per iteration, representing an average of 25 daily fishing attempts. This study preserves this repetition count exactly, applying the same linearly decaying-amplitude, multiplicative, greedy-accepting repeated local-search mechanism as in the original publication.

Rather than adopting a generic approach that represents only the conceptual exploration-exploitation logic of the three phases, this adaptation preserves the original authors' mathematical formulation as-is, including the coefficient definitions, decaying envelopes, multi-source attraction structure, selection rule, and repetition schedule. The only difference is that the search space is relocated from the engineering design bounds to the $[0, 1)$ random-key interval. The method is therefore referred to in this study as DR-GBFIO.

5.3 Constraint Management

The system constraints embedded in the proposed model are structurally managed at the encoding and decoding layers, without requiring the complex penalty functions that constitute a common handicap in the metaheuristic optimization literature. This design choice, achieved through the use of these constraints, completely prevents infeasible solutions from infiltrating the population during the search process, thereby guaranteeing that the population produces only valid solutions at every iteration and directly increasing the algorithm's convergence efficiency.

5.3.1 Hard Constraints

These are the capacity and feasibility constraints. Hard constraints related to build-volume limits and hardware material support are handled directly by the filtering mechanism in the decoding stage. Because assignments are made only through the pre-computed pool of feasible printers, there is no need to define a separate penalty function. This ensures that every individual in the population offers an always-feasible encoding structure that inherently represents a valid schedule.

5.3.2 Operational Constraints

These are the maintenance and timing constraints. Dynamic operational constraints such as mandatory preventive-maintenance windows and sequence-dependent setup times are likewise managed by being automatically incorporated into the schedule via an event-driven scheduler during the decoding stage.

5.4 Benchmark Algorithms

The performance of the DR-GBFIO proposed in this study is compared against three benchmark algorithms considered significant in the literature. These algorithms are: Particle Swarm Optimization (PSO) [21], a classical continuous-space baseline run with the same random-key encoding; Genetic Algorithm (GA) [22], a classical permutation-friendly baseline incorporating arithmetic crossover, Gaussian mutation, and elitism; and Harris Hawks Optimization (HHO) [19], a nature-inspired algorithm featuring an energy-decay-based exploration/exploitation transition and Levy-flight-based local search. All algorithms were run with the same

encoding, the same decoding function, the same population size (30), and the same number of iterations (80) to ensure a fair comparison. The internal hyperparameters of each benchmark algorithm (PSO's inertia weight and acceleration coefficients; GA's crossover and mutation probabilities) were set to the canonical values recommended in their respective seminal publications rather than tuned specifically for this problem, since HHO does not require problem-specific control parameters comparable to the PSO and GA settings considered here, and a tuning procedure applied to only some of the algorithms would compromise the fairness of the comparison.

The original, continuous-space GBFIO [3] is not included as a separate baseline because it cannot be applied to the discrete assignment-and-sequencing decisions studied here without first being encoded, which is precisely the contribution this study provides. Comparing DR-GBFIO against a continuous optimizer operating on a fundamentally different search space would not constitute a meaningful baseline. Improved variants of GBFIO are likewise unavailable for comparison, as the algorithm was proposed in 2025 and no such variants have yet appeared in the literature.

5.5 Computational Complexity

Let n denote the number of jobs (the dimension of the random-key vector), m the number of printers, P the population size, T the maximum number of iterations, and R the number of Phase-3 local-search repetitions. Within a single iteration, Phase 1 updates each of the P individuals using the five current best individuals, identified via an $O(P \log P)$ sort, at a cost of $O(n)$ per individual, giving $O(P \log P + Pn)$. Phase 2 updates each individual with an $O(n)$ hunt step and an $O(n)$ care step, giving $O(Pn)$. Phase 3 applies R repeated $O(n)$ local updates to each individual, giving $O(PRn)$ and dominating the per-iteration cost since $R > 1$.

Each candidate solution generated in any phase is evaluated by the decoder, which assigns jobs to printers in $O(n)$ time using the pre-computed feasibility matrix, sorts the jobs queued on each printer by their sequencing key in $O(n \log n)$ time overall, and constructs the event-driven timeline, including maintenance-block insertion, in a further $O(n)$ traversal; the decoder therefore runs in $O(n \log n)$, dominated by the sequencing sort. Combining the number of evaluations performed per iteration ($O(PR)$, dominated by Phase 3) with the per-evaluation decoding cost gives an overall time complexity of $O(TPRn \log n)$ for DR-GBFIO. The space complexity is $O(n(P + m))$, reflecting the storage of the population matrix (P individuals of dimension n) together with the per-printer job queues and the precomputed feasibility matrix.

The algorithm is implemented in Python 3 using vectorized NumPy operations for the phase updates, with the instance generator, the four optimization algorithms, and the event-driven decoder organized as separate, independently testable modules; combined with deterministic, hash-based random-seed generation, this design supports exact reproducibility of all reported results.

6 Experimental Design

6.1 Instance Scales and Generation Rules

At three problem scales (Small: 5 printers $\times$ 20 jobs; Medium: 10 printers $\times$ 50 jobs; Large: 20 printers $\times$ 100 jobs), 10 independent random *instances* were generated for each scale. This choice ensures that algorithms do not exhibit overfitting to a specific instance, which is common in the metaheuristic scheduling literature; the number of instances was determined by balancing computational budget against statistical diversity. The instance-generation rules are summarized in [Table 3](#).

Table 3: Instance generation rules and parameter value ranges.

Parameter Property	Symbol	Value Range/Probability Distribution
Processing time	p_j	$\sim U(30, 480)$ min
Material distribution	–	50% PLA, 25% ABS, 25% PETG
Color distribution	–	Uniform (equal) probability across 5 different colors
Part dimensions	b_j	Between 20% and 90% of typical build volume
Maintenance interval	I_i	$\sim U(40, 100)$ h
Maintenance duration	D_i^{maint}	$\sim U(30, 90)$ min
Speed coefficient	v_i	$\sim U(0.8, 1.3)$
Energy consumption rate	e_i	$\sim U(0.15, 0.35)$ kW
Material support	M_i	Random subset of 1 to 3 for each printer

Because each of the four algorithms is stochastic, a single run on a given instance reflects one particular random-number trajectory rather than the algorithm's typical behavior on that instance; 10 independent runs are therefore performed per instance, differing only in random seed while the printer and job configuration remain fixed, so that the mean and variability of an algorithm's performance on that instance can be estimated rather than inferred from a single, potentially unrepresentative outcome. For each *scale* $\times$ *instance* $\times$ *algorithm* combination, 10 independent runs were performed, and in total 3 scales $\times$ 10 instances $\times$ 4 algorithms $\times$ 10 runs = 1200 independent optimization runs were executed. This design provides 100 observations (10 instances $\times$ 10 runs) for each algorithm-scale combination and reduces the risk of pseudoreplication in the subsequent statistical analysis, which treats the instance rather than the individual run as the independent unit. All algorithms were run with a population size of 30 and a maximum of 80 iterations. The instance/run generation arising from randomness was seeded with a deterministic hash function (Secure Hash Algorithm 256-bit, SHA-256) for reproducibility purposes. This allows the codebase to reproduce the reported results exactly, independent of Python's default per-process randomized *string hashing* behavior.

6.2 Performance Metrics and Statistical Tests

The study's primary performance metric was defined as the maximum completion time (*makespan*, minutes). In addition, average printer utilization rate, total energy consumption, and total setup time were monitored as secondary performance indicators.

Because the 10 runs performed on a given instance share the same underlying problem configuration and are therefore not statistically independent of one another, the instance, rather than the individual run, is treated as the independent unit of analysis for hypothesis testing. For each algorithm-instance pair, the mean makespan over the 10 runs is first computed, yielding 10 independent observations per algorithm at each problem scale (30 when all scales are combined). The non-parametric Friedman test was applied to these instance-level means for the statistical analysis of overall performance differences among the optimization algorithms, both separately for each problem scale and with all scales combined. For pairwise comparisons between algorithms, the *Wilcoxon signed-rank test* was applied to the same instance-level means, and the matched-pairs rank-biserial correlation, $r = (n_+ - n_-)/(n_+ + n_-)$ with n_+ and n_- denoting the number of instances favoring each algorithm, was computed to measure the effect size of the statistical differences. The significance level for all statistical tests was set at $\alpha = 0.05$. Descriptive statistics over the full set of 1200 runs, for which the independence assumption does not apply, are reported separately below.

7 Results and Discussion

7.1 Descriptive Statistics

All tests were performed single-threaded on a Python 3-based cloud computing environment; therefore, the reported CPU times may vary across different hardware and should be interpreted for relative comparison between algorithms. The mean, standard deviation, best (minimum) makespan, and average CPU time computed over 100 observations (10 instances $\times$ 10 runs) for each scale and algorithm are shown in [Table 4](#).

Table 4: Makespan statistics (minutes) and average CPU time (seconds) by scale and algorithm.

Scale	Algorithm	Mean Makespan	Std. Dev.	Best (min)	Mean CPU (s)
Small (5×20)	DR-GBFIO	1181.4	424.8	836.9	8.60
	PSO	1235.2	413.1	842.3	0.36
	GA	1229.0	420.1	839.4	0.45
	HHO	1234.2	409.8	859.2	0.76
Medium (10×50)	DR-GBFIO	1443.8	116.8	1217.2	18.14
	PSO	1559.1	144.5	1273.5	0.80
	GA	1566.2	138.8	1293.0	0.89
	HHO	1747.6	177.8	1376.7	1.61
Large (20×100)	DR-GBFIO	1671.1	89.0	1506.6	33.54
	PSO	1746.7	134.7	1423.8	1.24
	GA	1831.0	106.7	1609.3	1.30
	HHO	2049.7	173.7	1640.2	2.37

As shown in [Table 4](#), DR-GBFIO achieved the lowest (best) mean maximum completion time (makespan) at all three problem scales. At the small scale (5×20), DR-GBFIO outperformed GA (1229.0 min), HHO (1234.2 min), and PSO (1235.2 min) with a mean of 1181.4 min; this difference is statistically significant.

At the medium scale (10×50), DR-GBFIO showed a clear advantage over PSO (1559.1 min) and GA (1566.2 min), with a mean of 1443.8 min, while HHO (1747.6 min) exhibited by far the weakest performance.

At the large scale (20×100), DR-GBFIO (1671.1 min) achieved a clear advantage over PSO (1746.7 min), GA (1831.0 min), and particularly HHO (2049.7 min). This performance pattern shows that DR-GBFIO, applied faithfully to the original equations, exhibits a consistent and strong search behavior across all three problem scales as a result of its multi-source attraction structure, linearly decaying exploration/exploitation coefficients, and repeated local-intensification mechanism.

7.2 Convergence and Distribution Behavior

The convergence plots, averaged over 10 independent runs on a representative instance for each problem scale, are shown in [Fig. 1](#).

As shown in [Fig. 1](#), DR-GBFIO exhibits faster and deeper convergence than GA and HHO at all three problem scales, and faster convergence than PSO at the small and medium scales. At the large scale, DR-GBFIO and PSO exhibit comparable convergence behavior on this representative instance, consistent with the smaller effect size observed between these two algorithms once all instances are pooled together. This can

be explained by the combined effect of the multi-source attraction structure in Phase 1 and the repeated local-intensification mechanism in Phase 3, as the algorithm preserves population diversity while simultaneously conducting intensive local search in promising regions. HHO, on the other hand, appears to stagnate early, particularly at the medium and large scales, and converges more slowly than its competitors. This observation suggests that the Lévy-flight-based local search mechanism may not provide the expected effect in the random-key-encoded scheduling search space. The makespan distributions of the algorithms across problems are shown in Fig. 2.

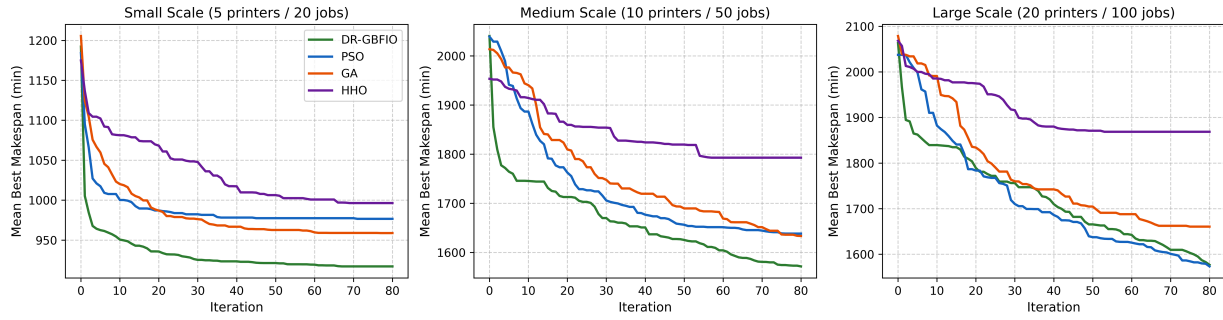


Figure 1: Convergence curves of the algorithms across three problem scales, averaged over 10 independent runs.

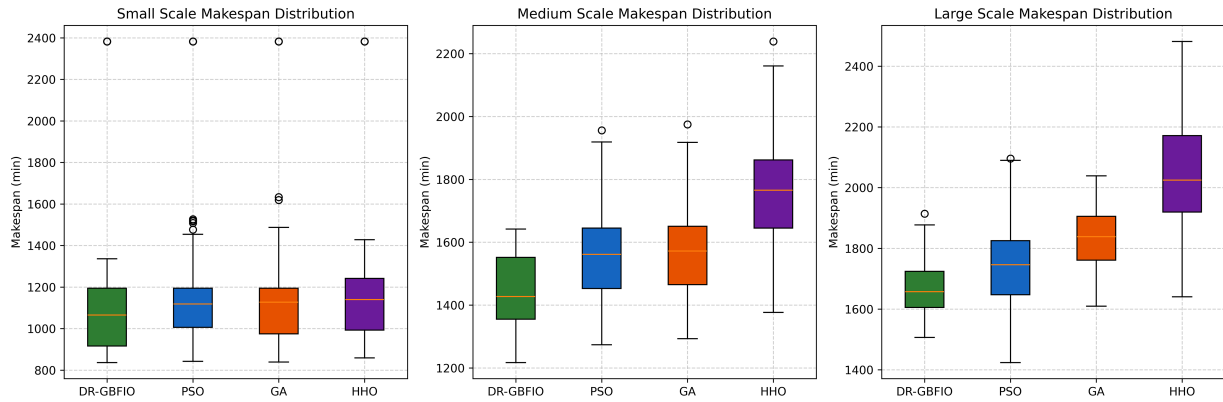


Figure 2: Makespan distributions of the algorithms, 100 observations/scale (10 instances × 10 runs).

The box plots presented in Fig. 2 reveal that DR-GBFIO produces not only better mean performance but also more stable results in terms of the spread of the solutions obtained (particularly at the medium and large scales). HHO, by contrast, is consistently the weakest and most unstable algorithm both in terms of mean performance and variability across independent runs.

7.3 Secondary Performance Indicators: Utilization Rate, Energy, and Setup Time

The average printer utilization rate, total energy consumption, and total setup times computed over the best solutions found are presented in Table 5.

Printer utilization rate and total energy consumption remain at relatively similar levels across the four algorithms. This is an expected finding, since the total workload is fixed, so the utilization rate depends primarily on the makespan/total-processing-time ratio and varies only to a limited extent independent of algorithm choice. In contrast, DR-GBFIO achieves the lowest value at all three scales for the total setup-time indicator, at 147.8 min at the small scale, 436.2 min at the medium scale, and 957.2 min at the large scale. This

indicates that the job sequences found by DR-GBFIO consistently organize material- and color-change costs more efficiently than the other three algorithms. This pattern is an important secondary finding indicating that DR-GBFIO's superiority in makespan performance stems not only from better printer assignments but also from finding job sequences that generate lower sequence-dependent setup costs.

Table 5: Average printer utilization rate, energy consumption, and total setup time by scale and algorithm.

Scale	Algorithm	Mean Utilization Rate	Mean Energy (kWh)	Mean Total Setup (min)
Small (5×20)	DR-GBFIO	96.9%	19.6	147.8
	PSO	96.6%	19.6	161.3
	GA	96.6%	19.6	159.0
	HHO	96.7%	19.6	155.9
Medium (10×50)	DR-GBFIO	96.7%	52.6	436.2
	PSO	96.5%	52.7	476.1
	GA	96.5%	52.7	477.9
	HHO	96.6%	52.7	465.6
Large (20×100)	DR-GBFIO	96.4%	99.9	957.2
	PSO	96.3%	100.0	979.0
	GA	96.3%	100.2	989.5
	HHO	96.3%	100.4	1004.0

7.4 Statistical Significance Analysis

The Friedman test results and mean ranks (lowest = best) for each scale are shown in [Table 6](#).

Table 6: Friedman test results and mean ranks by scale.

Scale	χ^2 (Friedman)	p -Value	RankDR-GBFIO	Rank PSO	Rank GA	Rank HHO
Small (5×20)	15.45	0.0015	1.30	3.10	2.60	3.00
Medium (10×50)	27.12	<0.001	1.00	2.60	2.40	4.00
Large (20×100)	28.92	<0.001	1.10	1.90	3.00	4.00

The difference among the algorithms was found to be statistically significant at all three problem scales ($p < 0.01$ at the small scale, $p < 0.001$ at the medium and large scales). DR-GBFIO achieved the best mean rank at all scales evaluated. At the small scale, with a mean rank of 1.30, it created a clear gap compared to the second-ranked GA (2.60); the small-scale statistic reflects a tie-corrected Friedman test, since all four algorithms reached an identical mean makespan on 2 of the 10 instances, which are small and low-complexity enough to admit a common optimal or near-optimal schedule. At the medium scale, with a value of 1.00, DR-GBFIO obtained the best rank in every one of the 10 instances, a clear gap ahead of the second-ranked GA (2.40). At the large scale, it again ranked first with a value of 1.10, maintaining a clear gap over the second-ranked PSO (1.90). HHO, on the other hand, consistently exhibited the worst performance at all three scales (with mean ranks of 3.00, 4.00, and 4.00, respectively). This pattern indicates that the performance superiority exhibited by DR-GBFIO is not coincidental and is consistently maintained, and in fact becomes more pronounced, across all problem sizes.

The results and interpretations of the pairwise Wilcoxon signed-rank test between DR-GBFIO and each benchmark algorithm are shown in Table 7.

Table 7: Pairwise Wilcoxon comparison of DR-GBFIO with other algorithms, by scale.

Scale	Comparison	W	<i>p</i> -Value	Rank-Biserial <i>r</i>	Interpretation
Small (5×20)	PSO	0	0.0078	1.000	DR-GBFIO significantly superior
	GA	0	0.0078	1.000	DR-GBFIO significantly superior
	HHO	0	0.0078	1.000	DR-GBFIO significantly superior
Medium (10×50)	PSO	0	0.0020	1.000	DR-GBFIO significantly superior
	GA	0	0.0020	1.000	DR-GBFIO significantly superior
	HHO	0	0.0020	1.000	DR-GBFIO significantly superior
Large (20×100)	PSO	1	0.0039	0.800	DR-GBFIO significantly superior
	GA	0	0.0020	1.000	DR-GBFIO significantly superior
	HHO	0	0.0020	1.000	DR-GBFIO significantly superior

The overall Wilcoxon results with all scales combined ($n = 30$ paired instance-level observations) are presented in Table 8. The corresponding overall Friedman mean ranks across all scales combined are shown in Fig. 3.

Table 8: Overall Wilcoxon comparison of DR-GBFIO and other algorithms with all scales combined.

Comparison	W Statistic	<i>p</i> -Value
DR-GBFIO vs. PSO	1	<0.001
DR-GBFIO vs. GA	0	<0.001
DR-GBFIO vs. HHO	0	<0.001

The four mean ranks plotted in Fig. 3 are computed from the unrounded per-instance ranks and sum to exactly 10 (that is, $k(k+1)/2$ for $k = 4$ algorithms) before rounding; the values displayed to two decimal places may deviate from this sum by a hundredth of a rank due to independent rounding for plotting purposes.

The experimental findings obtained offer three main takeaways. The first is that DR-GBFIO exhibits a consistent and complete performance superiority over all three benchmark algorithms across all three problem scales. DR-GBFIO significantly outperforms PSO, GA, and HHO at the small scale, the medium scale, and, notably, the large scale as well, including against PSO ($p = 0.0039$, $r = 0.800$). This indicates that applying the discrete random-key structure without altering the original mechanisms—such as the multi-source attraction structure, linearly decaying $\alpha/\rho/\gamma$ coefficient envelopes, and repeated local intensification—plays a decisive role in solution quality across the full range of problem sizes tested. In this respect, the findings offer a methodological reference regarding the importance of remaining faithful to the original mathematical formulation, including its full repetition schedule, when making improvements in algorithm-adaptation studies.

The second takeaway concerns scale-dependent effect size. While DR-GBFIO's advantage over PSO remains statistically significant and large at every scale, its magnitude, measured by the rank-biserial correlation, is slightly smaller at the large scale ($r = 0.800$) than at the small and medium scales ($r = 1.000$ at

both). This indicates that PSO's simple swarm-based update rule becomes relatively more competitive as the search space grows in dimensionality, even though it does not close the gap. This observation is consistent with the No-Free-Lunch theorem's expectation that no single algorithm's structural advantage is unbounded across problem scales, while still confirming DR-GBFIO's overall superiority within the tested range.

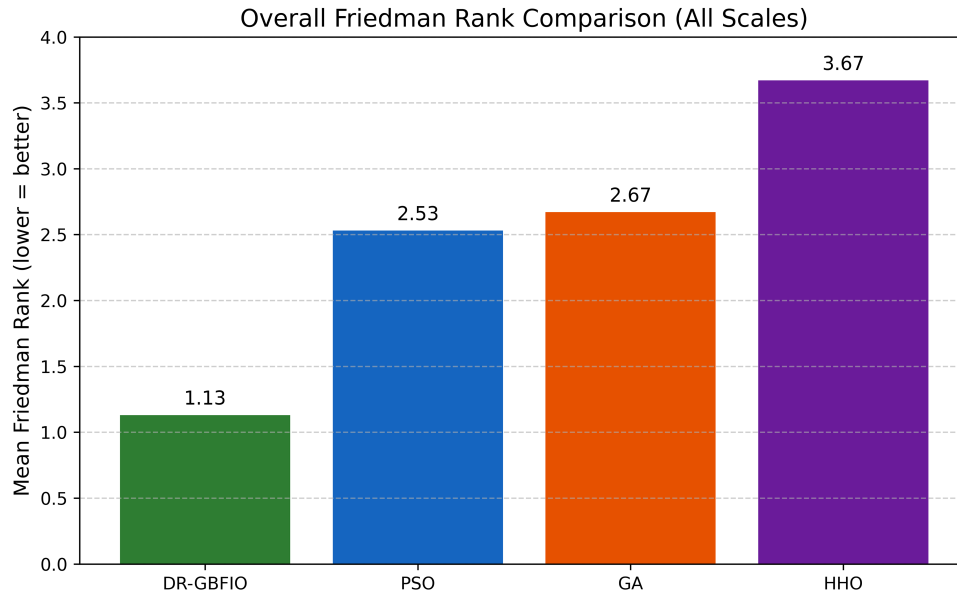


Figure 3: Overall Friedman mean ranks with all scales combined.

The final takeaway concerns search-space and operator mismatch. DR-GBFIO outperforms HHO with the maximum possible effect size ($r = 1.000$) at every scale, meaning DR-GBFIO's instance-level mean makespan was lower than HHO's on every single instance tested. HHO consistently exhibited the weakest performance across all scales, particularly for medium- and large-sized problems, suggesting that the Levy-flight-based local search mechanism, as implemented in HHO, may not be well suited to the topology of random-key-encoded discrete scheduling search spaces.

7.5 Limitations

This study has several limitations that should be considered. First, the experiments were conducted on synthetically generated instances. A validation stage using field data collected from an actual FDM farm would therefore strengthen the external validity of the findings obtained. Second, the current experimental setup is limited to the single-objective maximum completion time (makespan) model; the multi-objective weighted model given in Eq. (13) was not tested experimentally. As a third limitation, the population size and maximum iteration count parameters were fixed based on a practical trade-off between computation time and solution quality, and a systematic sensitivity analysis of these hyperparameters was not performed. Finally, although 10 independent random instances were generated for each problem scale to ensure statistical validity in the experimental design, increasing this number to 20 or 30 in future work has the potential to further increase the power of the statistical tests and the generalizability of the results.

7.6 Future Work

Considering the limitations of the current model and the findings obtained, future work can be shaped along several core directions. First, developing a multi-objective Pareto-based DR-GBFIO framework (for

example, hybridized with the Non-dominated Sorting Genetic Algorithm II (NSGA-II)) and experimentally testing the weighted model given in Eq. (13) are planned. To increase practical applicability in manufacturing processes, the goal is to incorporate additional industrial constraints into the model, such as printer availability rates, nozzle wear, and time-dependent energy pricing, and to address part placement (nesting) and production scheduling through an integrated approach. In addition to these developments, validating the proposed model with field data collected from an actual workshop (maker-lab) environment to test its practical validity will be a critical step.

8 Conclusion

In this study, a mathematical scheduling model was developed that jointly addresses material change, color change, build-volume and material feasibility, and preventive-maintenance constraints in heterogeneous FDM printer farms. The recently introduced GBFIO algorithm was adapted to this complex discrete problem while preserving its original equations exactly, using a random-key encoding approach. Statistically validated experimental analyses, conducted at three different problem scales (small, medium, and large) with four algorithms, 10 independent instances per scale, and a total of 1200 optimization runs, showed that DR-GBFIO achieves a statistically significant superiority over all three benchmark algorithms across all three problem scales.

The study's key contributions to the literature can be summarized as follows:

- Integrated formulation, in a manner consistent with industrial realities, of five operational elements rarely addressed together in the literature (heterogeneous printer fleet, material change, color change, preventive maintenance, and capacity constraints) within a single optimization model.
- Presentation of one of the first adaptations of the continuous-space-based GBFIO algorithm to combinatorial scheduling problems, and reporting of its performance through a statistically validated, large-scale experimental design.
- Empirical evidence that preserving DR-GBFIO's original phases (multi-source attraction, linearly decaying exploitation coefficients, repeated local intensification) and repetition schedule without approximation is associated with a stable and consistent performance superiority across all tested problem scales, although isolating the individual contribution of each mechanism would require a dedicated ablation study.

Acknowledgement: Not applicable.

Funding Statement: The author received no specific funding for this study.

Availability of Data and Materials: The datasets generated and analyzed during the current study, together with the source code used to generate and analyze them, are available from the corresponding author upon reasonable request.

Ethics Approval: Not applicable. This study did not involve human participants, human data, or animal subjects; all experiments were conducted on synthetically generated problem instances.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Garey MR, Johnson DS. Computers and intractability: a guide to the theory of NP-completeness. New York, NY, USA: W. H. Freeman and Company; 1979.
2. Pinedo ML. Scheduling: theory, algorithms, and systems. 6th ed. Cham, Switzerland: Springer International Publishing; 2022. doi:10.1007/978-3-031-05921-6.

3. Dehghani M, Aly M, Rodriguez J, Sheybani E, Javidi G. A novel nature-inspired optimization algorithm: grizzly bear fat increase optimizer. *Biomimetics*. 2025;10(6):379. doi:10.3390/biomimetics10060379.
4. Li Q, Kucukkoc I, Zhang DZ. Production planning in additive manufacturing and 3D printing. *Comput Oper Res*. 2017;83:157–72. doi:10.1016/j.cor.2017.01.013.
5. Chergui A, Hadj-Hamou K, Vignat F. Production scheduling and nesting in additive manufacturing. *Comput Ind Eng*. 2018;126:292–301. doi:10.1016/j.cie.2018.09.048.
6. Kucukkoc I. MILP models to minimise makespan in additive manufacturing machine scheduling problems. *Comput Oper Res*. 2019;105:58–67. doi:10.1016/j.cor.2019.01.006.
7. Kim YJ, Kim BS. Part-grouping and build-scheduling with sequence-dependent setup time to minimize the makespan for non-identical parallel additive manufacturing machines. *Int J Adv Manuf Technol*. 2022;119(3):2247–58. doi:10.1007/s00170-021-08361-z.
8. Oh Y, Witherell P, Lu Y, Sprock T. Nesting and scheduling problems for additive manufacturing: a taxonomy and review. *Addit Manuf*. 2020;36(10):101492. doi:10.1016/j.addma.2020.101492.
9. Pinto M, Silva C, Thürer M, Moniz S. Nesting and scheduling optimization of additive manufacturing systems: mapping the territory. *Comput Oper Res*. 2024;165(1):106592. doi:10.1016/j.cor.2024.106592.
10. Ransikarbum K, Ha S, Ma J, Kim N. Multi-objective optimization analysis for part-to-printer assignment in a network of 3D fused deposition modeling. *J Manuf Syst*. 2017;43:35–46. doi:10.1016/j.jmsy.2017.02.012.
11. Almasarwah N. Optimized green unrelated parallel machine scheduling problem subject to preventive maintenance. *Designs*. 2025;9(2):26. doi:10.3390/designs9020026.
12. Ying KC, Pourhejazy P, Huang YH. Tailored iterated greedy metaheuristic for a scheduling problem in metal 3D printing. *Adv Eng Softw*. 2023;186:103546. doi:10.1016/j.advengsoft.2023.103546.
13. Avalos-Rosales O, Angel-Bello F, Álvarez A, Cardona-Valdés Y. Including preventive maintenance activities in an unrelated parallel machine environment with dependent setup times. *Comput Ind Eng*. 2018;123:364–77. doi:10.1016/j.cie.2018.07.006.
14. Durašević M, Jakobović D. Heuristic and metaheuristic methods for the parallel unrelated machines scheduling problem: a survey. *Artif Intell Rev*. 2023;56(4):3181–289. doi:10.1007/s10462-022-10247-9.
15. Zhang W, Bao X, Hao X, Gen M. Metaheuristics for multi-objective scheduling problems in industry 4.0 and 5.0: a state-of-the-arts survey. *Front Ind Eng*. 2025;3:1540022. doi:10.3389/fieng.2025.1540022.
16. Wang L, Qi Y. Scheduling an energy-aware parallel machine system with deteriorating and learning effects considering multiple optimization objectives and stochastic processing time. *Comput Model Eng Sci*. 2023;135(1):325–39. doi:10.32604/cmesci.2022.019730.
17. Wang C, Li X, Gao Y. A novel collaborative evolutionary algorithm with two-population for multi-objective flexible job shop scheduling. *Comput Model Eng Sci*. 2023;137(2):1849–70. doi:10.32604/cmesci.2023.028098.
18. Zhang H, Chen Y, Zhang Y, Xu G. Energy-saving distributed flexible job shop scheduling optimization with dual resource constraints based on integrated Q-learning multi-objective grey wolf optimizer. *Comput Model Eng Sci*. 2024;140(2):1459–83. doi:10.32604/cmesci.2024.049756.
19. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H. Harris Hawks optimization: algorithm and applications. *Future Gener Comput Syst*. 2019;97:849–72. doi:10.1016/j.future.2019.02.028.
20. Bean JC. Genetic algorithms and random keys for sequencing and optimization. *ORSA J Comput*. 1994;6(2):154–60. doi:10.1287/ijoc.6.2.154.
21. Kennedy J, Eberhart R. Particle swarm optimization. *Proc IEEE Int Conf Neural Netw*. 1995;4:1942–8.
22. Holland JH. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. Cambridge, MA, USA: The MIT Press; 1992. doi:10.7551/mitpress/1090.001.0001.