Time-Aware Two-Dimensional Packing for Slicing-Aware 3D Printing Throughput Optimization

Stephone Christian[1], **Blayne Montaque**[2], **Saurav K. Aryal**[2*]

[1]Stanford University
[2]Howard University
stephone@stanford.edu, blayne.montaque@bison.howard.edu, saurav.aryal@howard.edu

Abstract

Batching multiple parts onto a single fused-filament fabrication build plate improves throughput, but existing packing algorithms optimize for geometric density rather than print time. We introduce a slicing-aware cost model that estimates print time from part geometry and placement without invoking toolpath generation, achieving strong correlation with slicer-reported times (Pearson $r = 0.90$, Spearman $\rho = 0.96$). Evaluating packing algorithms on synthetic production builds, we find that greedy polygon-based packing matches or exceeds Large Neighborhood Search at three orders of magnitude lower compute cost—a negative result we attribute to high initial packing density leaving minimal room for iterative refinement. Against PrusaSlicer's default auto-arrange, our method achieves 5.7% throughput improvement (95% CI [4.7%, 7.3%], $N = 329$), with median savings of 19.5 minutes per build.

Introduction

Additive manufacturing workflows frequently require producing multi-part assemblies where each assembly contains a bill of materials (BOM) specifying required quantities of distinct parts. A standard approach prints parts sequentially as independent jobs, incurring repeated job-level overhead such as heating, homing, and priming. Packing multiple parts on a single build plate can reduce this overhead by amortization. However, practitioners often observe that the speedup from batching exceeds overhead amortization alone. Contemporary slicers commonly generate toolpaths that exploit cross-part structure within a layer, including global ordering of disjoint "islands" and longer contiguous extrusion segments. These effects can reduce travel and increase effective extrusion efficiency (Liu et al. 2020).

This paper develops a formulation in which packing decisions are guided by a slicing-aware estimate of print time. The objective is throughput over a finite horizon, measured as the expected number of complete assemblies produced per unit time. The key modeling choice is to treat print time as a function of (i) the selected part multiplicities, (ii) the corresponding feasible two-dimensional placement, and (iii) a slicer-motivated approximation of travel between islands across layers.

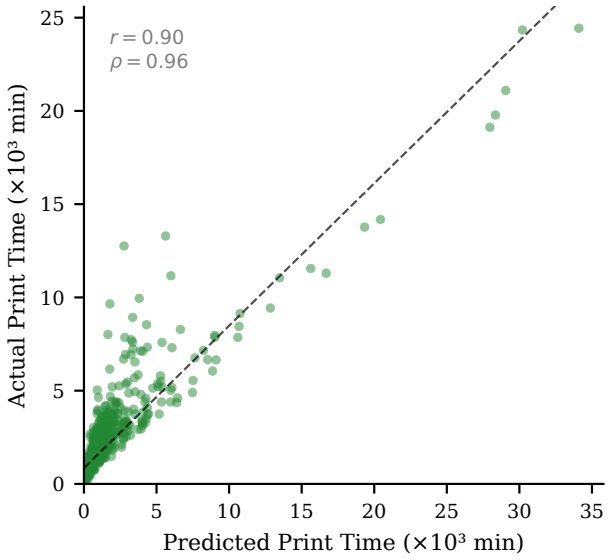


Figure 1: Predicted vs. actual print time for $N = 1{,}000$ builds. Actual times are PrusaSlicer estimates derived from generated toolpaths. Dashed line shows linear fit ($r = 0.90$, $\rho = 0.96$).

Related Work

Two-dimensional packing and bin packing have a long history, with many heuristics and metaheuristics targeting area utilization or number of bins. Separately, toolpath planning and travel minimization problems are often related to routing and tour construction over spatial regions. In additive manufacturing, throughput modeling has typically emphasized machine parameters, material deposition rates, and per-job overhead, while production planning approaches often abstract away slicer-dependent toolpath effects (Oh et al. 2020).

The present work connects these themes by (i) using packing not only to fit parts but also to expose a placement-induced structure (island centers) for travel estimation, and

*Corresponding author.

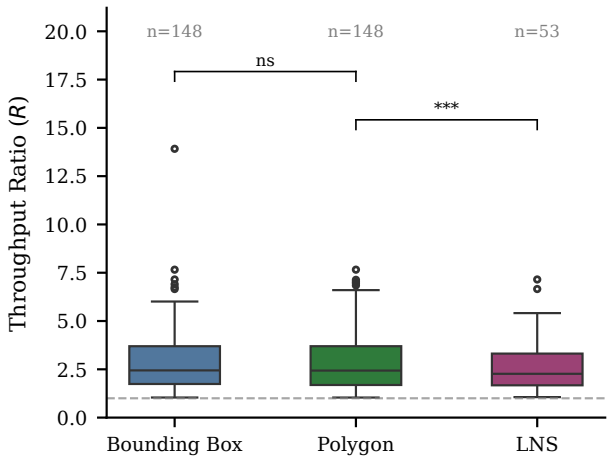


Figure 2: Distribution of throughput ratios by algorithm. Polygon Projection and Bounding Box achieve similar distributions; LNS shows slightly lower throughput despite higher compute cost (Phase 2; $N = 148$ Polygon/BB, $N = 53$ for LNS).

(ii) optimizing part multiplicities and packing jointly under a horizon-based throughput objective rather than a purely geometric metric.

Problem Formulation

Consider a build consisting of P unique part types. Part i is characterized by its planar footprint $F_i \subset \mathbb{R}^2$, height H_i, and nominal extruded volume V_i. Let $x_i \in \mathbb{Z}_{\geq 0}$ denote the number of copies of part i placed in a single job.

The build plate is a rectangle $S = [0, W] \times [0, D]$. A packing is feasible if there exist translations (u_{ik}, v_{ik}) for each copy $k \in \{1, \ldots, x_i\}$ such that

$$F_i + (u_{ik}, v_{ik}) \subseteq S, \tag{1}$$

and for any two distinct placed copies $(i, k) \neq (j, \ell)$,

$$\big(F_i + (u_{ik}, v_{ik})\big) \cap \big(F_j + (u_{j\ell}, v_{j\ell})\big) = \emptyset. \tag{2}$$

A packing procedure returns feasible placements and associated island centers

$$C(x) = \{c_{ik} \in \mathbb{R}^2 : \text{center of the placed copy } (i, k)\}. \tag{3}$$

Assembly completion. Let m_i be the BOM requirement (copies of part i per assembly). One job producing counts x yields

$$K_{\text{job}}(x) = \min_i \left\lfloor \frac{x_i}{m_i} \right\rfloor \tag{4}$$

complete assemblies (builds).

Objective. Given a time horizon T and a one-time computation/setup cost $t^{(1)}_{\text{comp}}$ for selecting x and generating a packing, the expected number of jobs executable is

$$N_{\text{jobs}}(x) = \frac{T - t^{(1)}_{\text{comp}}}{\tau_{\text{print}}(x)}, \tag{5}$$

where $\tau_{\text{print}}(x)$ is the slicing-aware print-time estimate of a packed job. The total number of complete assemblies is

$$K_{\text{total}}(x) = N_{\text{jobs}}(x)\, K_{\text{job}}(x), \tag{6}$$

and throughput is

$$\theta(x) = \frac{K_{\text{total}}(x)}{T}. \tag{7}$$

We seek

$$\max_{x \in \mathbb{Z}^P_{\geq 0}} \ \theta(x) \quad \text{s.t. packing feasibility on } S. \tag{8}$$

Slicing-Aware Print-Time Model

We model print time as a sum of job overhead, extrusion time, and travel time:

$$\tau_{\text{print}}(x) = t_0 + t_{\text{ext}}(x) + t_{\text{trav}}(x), \tag{9}$$

where t_0 is a per-job constant (heating/homing/priming).

Extrusion-Time Model

Let the total nominal extruded volume be

$$V_{\text{ext}}(x) = \sum_{i=1}^{P} x_i V_i. \tag{10}$$

With line width w and layer height h, an equivalent total extruded path length is

$$L_{\text{ext}}(x) = \frac{V_{\text{ext}}(x)}{wh}. \tag{11}$$

Let v_{ext} be a nominal extrusion (feed) speed and $Q_{\max}$ a nominal volumetric flow limit. For packed jobs, slicer-induced effects are captured by scaling factors:

$$\begin{aligned} v^*_{\text{ext}} &= s_v\, v_{\text{ext}}, \quad Q^*_{\max} = s_Q\, Q_{\max}, \\ V^*_{\text{ext}}(x) &= c_V\, V_{\text{ext}}(x), \end{aligned} \tag{12}$$

with $s_v \geq 1$ (effective speed), $s_Q \geq 1$ (flow utilization), and $c_V \leq 1$ (reduced redundant extrusion). The packed-job extrusion time is modeled as

$$t_{\text{ext}}(x) = \max\left(\frac{L_{\text{ext}}(x)}{v^*_{\text{ext}}},\ \frac{V^*_{\text{ext}}(x)}{Q^*_{\max}} \right). \tag{13}$$

Travel-Time Approximation

Travel in packed jobs includes intra-island motion and inter-island motion. We approximate inter-island motion by using island centers derived from the packing.

Let $L_{\max} = \left\lceil \max_i \frac{H_i}{h} \right\rceil$ be the number of layers up to the tallest part. At layer ℓ, define the set of active islands

$$C_\ell(x) = \{c_{ik} \in C(x) : H_i \geq \ell h\}. \tag{14}$$

For each active layer, we approximate inter-island travel length by a nearest-neighbor tour over $C_\ell(x)$:

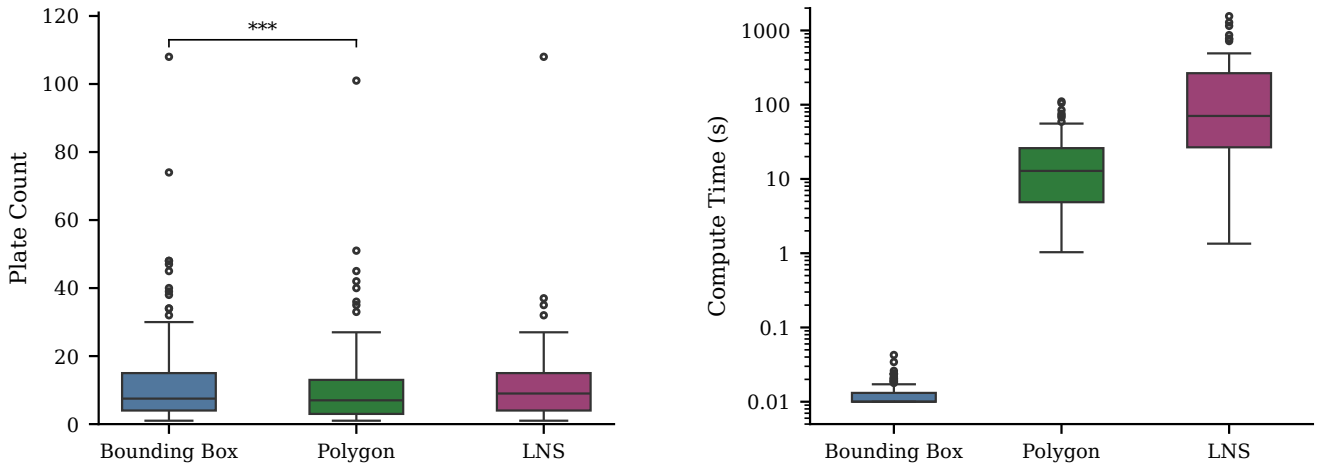
$$L_{\text{inter}}(C_\ell) \approx \text{NN-Tour}(C_\ell), \tag{15}$$

where NN-Tour denotes the path length produced by a greedy nearest-neighbor heuristic. Let $L_{\text{intra}}(C_\ell)$ denote intra-island travel aggregated across islands at layer ℓ. The total travel length is

$$L_{\text{trav}}(x) = \sum_{\ell=1}^{L_{\max}} \left(L_{\text{intra}}(C_\ell(x)) + L_{\text{inter}}(C_\ell(x)) \right), \tag{16}$$

and the travel time is

$$t_{\text{trav}}(x) = \frac{L_{\text{trav}}(x)}{v_{\text{trav}}}. \tag{17}$$



(a) Plate count

(b) Compute time

Figure 3: Operational metrics by algorithm. (a) Polygon packing achieves 13.5% fewer plates than Bounding Box ($p < 0.001$). (b) Bounding Box computes in <1s (median); Polygon Packing requires 13s (median); LNS requires 71s (median).

Throughput Analysis and Baseline Comparison

A sequential baseline prints each part type (or each individual part) in separate jobs. Let $t_{\text{ext}}^{\text{solo}}(i)$ and $t_{\text{trav}}^{\text{solo}}(i)$ denote estimated extrusion and travel time for printing part i in isolation with the same process settings. Then a representative baseline time for producing one set of parts can be written as

$$\tau_{\text{base}} = \sum_{i=1}^{P} \left(t_0 + t_{\text{ext}}^{\text{solo}}(i) + t_{\text{trav}}^{\text{solo}}(i) \right), \tag{18}$$

and baseline throughput as

$$\theta_{\text{base}} = \frac{1}{\tau_{\text{base}}}. \tag{19}$$

For a packed decision x, a horizon-adjusted improvement ratio is

$$R(x) = \frac{\theta(x)}{\theta_{\text{base}}} = \frac{T - t_{\text{comp}}^{(1)}}{T} \cdot \frac{K_{\text{job}}(x)}{\tau_{\text{print}}(x)} \cdot \tau_{\text{base}}. \tag{20}$$

Optimization Algorithm

The optimization couples discrete counts, packing feasibility, and a placement-dependent travel estimate. We use a nested strategy: an inner loop generates a feasible packing and island centers for a candidate x, and an outer loop searches over x to maximize $\theta(x)$ (or $R(x)$).

Methods

Dataset

Our experiments use STL meshes from Thingi10K, an open-source dataset of 10,000 models collected from Thingiverse, a popular 3D printing model-sharing platform (Zhou and Jacobson 2016). This dataset captures the geometric diversity and quality characteristics of real-world 3D printing workloads, including mechanical parts, toys, and decorative objects with varying complexity.

We filter the raw dataset to models suitable for the Bambu X1C build volume ($256 \times 256 \times 256$ mm), retaining meshes with maximum dimension between 10–256mm and effective volume above 100mm^3. After filtering, 4,187 meshes remain as candidates for build generation.

Build Generation

We generate synthetic builds to simulate production batch scenarios. Each build comprises a bill of materials (BOM) specifying P unique part types, where each part type i requires k_i copies. The number of unique parts is sampled uniformly:

$$P \sim \mathcal{U}(3, 15). \tag{21}$$

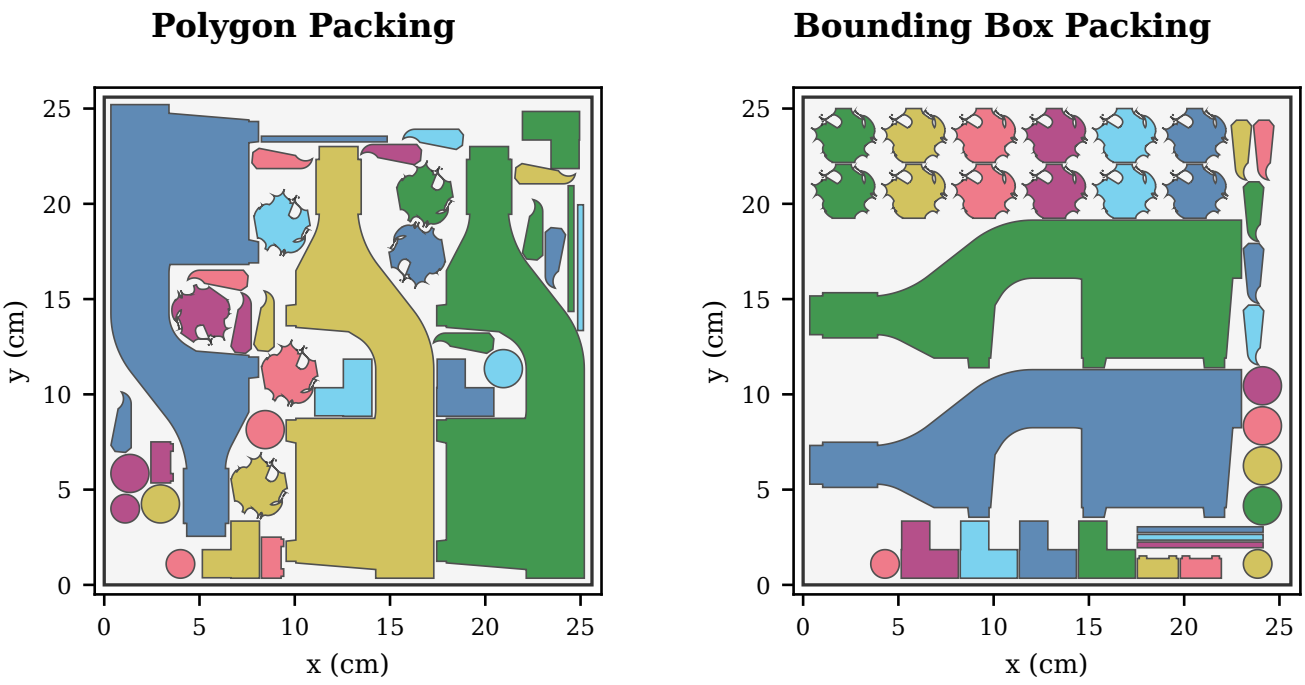
To reflect real-world production patterns where some parts are needed in larger quantities, copy counts follow a truncated log-normal distribution:

$$k_i \sim \min\big(\lceil \text{LogNormal}(\mu = 1, \sigma = 1) \rceil, 40\big). \tag{22}$$

Part selection uses popularity-weighted sampling without replacement, where popularity weights are drawn from a log-normal distribution to simulate the heavy-tailed usage patterns observed in manufacturing (i.e., a small number of parts appear frequently across builds). We generate $N = 2{,}000$ builds, yielding batches ranging from approximately 10 to 150 total parts per build after replication.

Mesh Orientation Preprocessing

Meshes from open-source repositories often lack consistent orientation conventions, as meshes can be made by different 3D softwares with different axis orderings. We employ



(a) Polygon Packing

(b) Bounding Box Packing

Figure 4: Example packing configurations produced by the Polygon Projection and the Bounding Box methods (projecting part geometry onto printer plate surface, *xy* axis)

a heuristic to reorient parts that are in orientations not conducive to additive manufacturing. We attempt to be conservative in our approach, as part orientation in additive manufacturing can be changed to influence the print quality of an object and its tensile strength.

Heuristic For each mesh, we compute stable poses using the convex hull analysis and score each candidate by:

$$S = w_c \cdot A_{\text{contact}} - w_o \cdot A_{\text{overhang}} - w_h \cdot z_{\text{COM}} \quad (23)$$

where A_{contact} is bed contact area, A_{overhang} is unsupported overhang area (faces with $n_z < -\cos 45°$), and z_{COM} is center-of-mass height. We use weights $w_c = 3$, $w_o = 10$, $w_h = 6$.

Validation We validate the heuristic on a stratified sample of 73 parts spanning three categories: "Big Win" (highest predicted improvement), "Regression" (predicted degradation), and "High Aspect Ratio" (elongated parts prone to tip-over). A human rater compared original and reoriented poses, voting $+1$ (better), 0 (equivalent), or -1 (worse)

The 88% safety rate (non-harmful mesh transformation) and high "same" (no change or equivalent quality mesh transformation) for high aspect ratio parts confirms the heuristic is conservative. Using a threshold of $\Delta > 10{,}000$, we reorient approximately 5% of parts in the full dataset.

Cost Function Model Validation

We validated our slicing-aware print-time model against empirical estimates from PrusaSlicer 2.7 (Prusa Research 2024). PrusaSlicer's toolpath generation algorithms are architecturally similar to those in Bambu Studio, both being descendants of the Slic3r codebase. This makes PrusaSlicer a suitable proxy for estimating print times on Bambu X1C hardware. Our empirical analysis consisted of $N = 1000$ builds.

Category	Better	Same	Worse
Big Win ($N = 25$)	72%	20%	8%
High AR ($N = 23$)	4%	91%	4%
Regression ($N = 25$)	16%	60%	24%
Overall ($N = 73$)	32%	56%	12%

Table 1: Orientation heuristic validation results

Packing Algorithms

Bounding Box Approximation Figure 4 shows a representative build packed by Polygon Projection and Bounding Box algorithms. Polygon's use of exact part footprints enables tighter interlocking arrangements, explaining the 13.5% reduction in plate count observed in our experiments.

Phase 3: Commercial Baseline Comparison

We use the MaxRects Bottom Left bin-packing heuristic (Jylänki 2010), implemented via the rectpack open source library (secnot 2017). Parts are approximated by axis-aligned bounding boxes, sorted by height or area, then placed greed-

Metric	Polygon	Prusa	Difference
Throughput Ratio (R)			
Mean ± SD	2.77 ± 1.42	2.61 ± 1.41	—
Median	2.52	2.35	—
Geo. mean improvement		—	$+5.7\%$
95% Bootstrap CI		—	$[+4.7\%, +7.3\%]$
Wilcoxon p		—	< 0.001
Plate Count			
Median	3	3	—
Mean	3.42	3.23	$+5.9\%$
Head-to-head wins	81 (24.6%)	29 (8.8%)	—
Wilcoxon p		—	0.016

Table 2: Phase 3: Polygon Projection vs. Prusa Auto-Arrange. Throughput ratio $R = \tau_{\text{seq}}/\tau_{\text{packed}}$. Four outliers removed ($R_{\text{Prusa}} < 0.5$); $N = 329$.

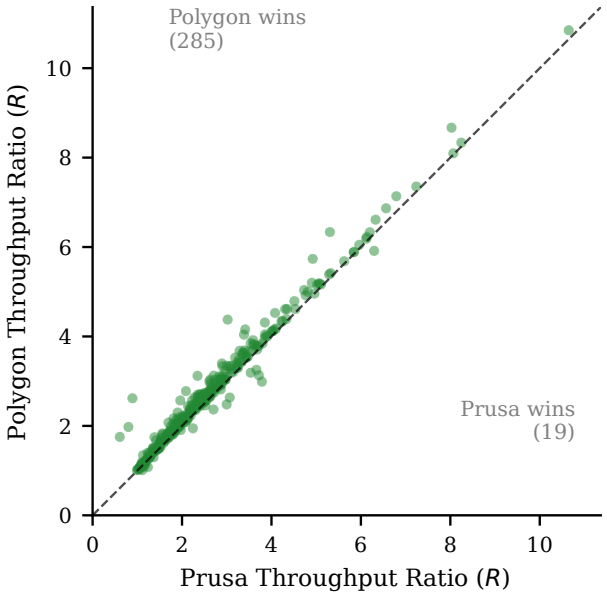


Figure 5: Phase 3: Polygon Projection vs. Prusa Auto-Arrange throughput ($N = 329$). Points above the diagonal indicate Polygon wins. Polygon achieves higher throughput on the majority of builds.

Algorithm 1: Time-Aware Packing Optimization

Input: Plate dimensions (W, D); part data $\{F_i, H_i, V_i, m_i\}_{i=1}^{P}$; horizon T; overhead t_0; process parameters $(w, h, v_{\text{ext}}, Q_{\max}, v_{\text{trav}})$; scaling factors (s_v, s_Q, c_V)

Output: Counts $x^\star$ and packing/centers for $x^\star$

Initialize $x \leftarrow x^{(0)}$; best $\leftarrow -\infty$
for iteration $= 1$ to N **do**
  Propose candidate counts $\tilde{x} \leftarrow \textsc{Neighbor}(x)$
  Attempt feasible packing of $\tilde{x}$ on S; obtain centers $C(\tilde{x})$
  if packing succeeds **then**
    Estimate $\tau_{\text{print}}(\tilde{x})$ using extrusion and travel models
    Compute $\theta(\tilde{x}) = \frac{T - t_{\text{comp}}^{(1)}}{T} \cdot \frac{K_{\text{job}}(\tilde{x})}{\tau_{\text{print}}(\tilde{x})}$
    if $\theta(\tilde{x}) >$ best **then**
      $x^\star \leftarrow \tilde{x}$; best $\leftarrow \theta(\tilde{x})$
    end if
    Accept/reject $\tilde{x}$ according to a metaheuristic rule; update x
  end if
end for
return $x^\star$ and its packing

ily with 90° rotation. Computation time varies from 10-300 ms.

Polygon Projection Parts are represented by their exact 2D footprint polygons, obtained by projecting the mesh geometry onto the xy-plane. The build plate is discretized into a binary occupancy grid at 1mm resolution. For each part, we precompute rasterized masks at multiple rotation angles (0°, 90°, 180°, 270°, and optionally 45° diagonals). Placement candidates are identified via FFT-accelerated convolution between the part mask and the current bed state, yielding all collision-free positions in $O(n \log n)$ time. Among valid positions, we select based on an adjacency score that favors placements near existing parts to improve packing density. A post-processing "gravity shake" phase iteratively removes and re-inserts parts to settle them into remaining gaps. After packing the plate, a final sweep is performed across smaller pieces to greedily pack remaining gaps. Computation time scales with part count and geometric complexity, typically 1–100 seconds for batches of 10-200 parts.

Large Neighborhood Search We implement a destroy-and-repair metaheuristic initialized from a Rectpack baseline. In each iteration, a *destroy phase* removes k plates (20% of current plates) selected by one of three criteria chosen randomly: tallest parts, lowest density, or highest print time. The *repair phase* reinserts removed parts greedily, placing each part on the plate that minimizes marginal cost increase (print time delta). A tabu list prevents recently-used

(part, plate) assignments for a tenure of 10–15 iterations, encouraging exploration of alternative configurations. The algorithm accepts improving moves unconditionally and uses an aspiration criterion to override tabu status when a new global best is found. We run for 20-50 iterations, requiring 3–10 minutes depending on problem size.

Experimental Design

Our experiments follow a three-phase structure: within-algorithm tuning (Phase 1), cross-algorithm comparison (Phase 2), and finally a comparison against Bambu's (PrusaSlicer's) native packing logic through the "arrange" command (Phase 3).

Phase 1: Within-Algorithm Tuning

- **Phase 1a** (Polygon Projection): We compare part sorting strategies, height-based vs area-based on $N = 192$ builds. Height-based sorting processes taller parts first, potentially reducing maximum plate height; area-based sorting prioritizes larger footprints for early placement.
- **Phase 1b** (Bounding Box): We evaluate height-based and area-based sorting strategies on $N = 176$ builds.
- **Phase 1c** (LNS): We compare two metaheuristic variants; Tabu search vs Tabu with plate consolidation on $N = 42$ builds due to computational constraints.

Phase 2: Cross-Algorithm Comparison The winning configuration from each Phase 1 experiment was compared head-to-head on common builds ($N = 53$ for three-way comparison, $N = 148$ for Polygon Projection vs Bounding Box). This design was imposed to balance statistical power against computational constraints. All experiments use a replica count $r = 3$, simulating production batches that require 3 copies of the complete Bill of Materials (BOM).

Phase 3: Bambu *Arrange* Baseline Comparison The winning packing algorithm from Phase 2 was compared to Bambu's "arrange" packing arrangement, $N = 333$. We use a replica count of $r = 1$ to satisfy computational constraints.

Statistical Analysis For each packing algorithm, the throughput ratio $R = \tau_{\text{seq}}/\tau_{\text{packed}}$ measures speedup relative to a sequential baseline where each part is printed as an independent job, incurring per-job overhead (heating, homing, priming) for every part. All comparisons used paired designs to control for build-specific variance. Throughput ratios were log-transformed prior to analysis, as ratios are multiplicatively distributed. For Phases 1–2, we used paired t-tests on log-transformed values and report geometric mean ratios with 95% bootstrap confidence intervals (10,000 resamples). Effect sizes are quantified using Cohen's d on log differences. Multiple comparisons used Holm–Bonferroni correction to control family-wise error at $\alpha = 0.05$.

For Phase 3 ($N = 329$), Shapiro–Wilk tests indicated non-normality in the paired log-differences ($W = 0.51$, $p < 0.001$), reflecting positive skew from builds where Polygon Projection substantially outperformed the baseline. We therefore confirmed results using the non-parametric Wilcoxon signed-rank test and report both geometric mean (+5.7%) and median (+3.6%) improvements with bootstrap confidence intervals. Four outliers with $R_{\text{Prusa}} < 0.5$ were excluded; sensitivity analyses confirmed results were robust to their inclusion.

GUI-Based Benchmarking of Prusa Auto-Arrange For Phase 3, PrusaSlicer's auto-arrange baseline was obtained through the PrusaSlicer GUI to ensure we evaluated the same placement logic used in practice. For each generated build, we loaded the STL set into PrusaSlicer, applied a fixed printer/process profile, and executed the built-in *Arrange* operation with identical spacing and exclusion settings across all builds. PrusaSlicer then produced a multi-plate arrangement when required.

To evaluate these arrangements under our cost model, we exported the arranged project as a multi-plate 3MF and extracted per-part placements directly from the 3MF contents. Specifically, we parsed the build-item transforms from `3D/3dmodel.model` to recover each instance's in-plane translation and rotation, and we inferred plate membership using Prusa's multi-plate metadata (wipe tower information) combined with data-driven clustering of item translations into distinct plate tiles. For each recovered plate, we constructed a packed-plate configuration (part list, centers, rotations) in the same format used by our packing algorithms and computed the print time using the our analytical cost function implementation.

Because the arrangement step was performed via GUI automation, precise wall-clock timing of Prusa's internal arrangement was not reliably measurable; we therefore treat Prusa compute time as qualitative (on the order of seconds) and focus Phase 3 primarily on throughput and cost-model comparisons derived from the recovered plate configurations.

Results

Cost Model Validation

The predicted cost showed a strong correlation with slicer-reported print time (Pearson $r = 0.90$, $p < 0.001$), indicating the model captures the dominant factors affecting print duration (Figure 1). Spearman rank correlation was $\rho = 0.96$, $p < 0.001$, confirming that the model is able to reliably rank alternative packings by print time, a critical property for optimization.

Phase1: Algorithm Configuration

Table 3 summarize within-algorithm tuning results. All pairwise comparisons were statistically significant ($p < 0.001$), however effect sizes varied substantially.

For Polygon Projection, height-based sorting marginally outperformed area-based, ($d = 0.33$). While statistically significant, this difference is unlikely to be operationally meaningful. For Bounding Box, height based sorting showed an improvement of 1.2%, $d = 0.46$). LNS with plate consolidation dramatically outperformed Tabu search (96% improvement).

Phase 2: Algorithm Comparison

Table 4 summarizes the head-to-head comparison of Phase 1 winners. Figure 2 shows throughput ratio distributions by

Algorithm	Configuration	R (mean $\pm$ std)	d	N
Polygon Projection	**Height-based sorting**	$\mathbf{2.73 \times \pm 1.51}$	0.33	192
	Area-based sorting	$2.71 \times \pm 1.49$	—	
Bounding Box	Area-based sorting	$2.95 \times \pm 2.16$	—	176
	Height-based sorting	$\mathbf{2.99 \times \pm 2.19}$	0.46	
LNS	Tabu	$1.57 \times \pm 1.03$	—	42
	Tabu Consolidation (LNS)	$\mathbf{3.08 \times \pm 1.90}$	> 2	

Table 3: Phase 1 results: Algorithm configuration tuning. Bold indicates selected configuration. Effect size d is Cohen's d on log-transformed ratios.

algorithm

Throughput Figure 2 shows that Polygon Projection and Bounding Box achieved statistically equivalent throughput ratios ($p = 0.93$, $d = 0.007$). Contrary to expectations, LNS underperformed Polygon Projection by 1.6% ($p < 0.001$, $d = 0.58$) despite the longer computation time.

Plate counts were log-transformed prior to analysis (Shapiro-Wilk $W = 0.65$ raw vs $W = 0.99$ log-transformed). Polygon Projection achieved 13.5% fewer plates than Bounding Box (geometric mean ratio $= 0.87$, 95% CI $[0.84, 0.91]$, $p < 0.001$, $d = -0.57$) and 11.2% fewer than LNS ($p < 0.001$, $d = -0.58$). The difference between Bounding Box and LNS was statistically significant but negligible (1.4%, $d = 0.32$).

Computational Efficiency

Figure 3b shows compute time by algorithm on a log scale. Bounding Box completed in under 1 second (median: <0.1s), Polygon required 13 seconds (median), and LNS required 71 seconds (median). The three-order-of-magnitude difference between Bounding Box and LNS reflects the fundamental tradeoff between greedy heuristics and metaheuristic search.

Notably, LNS's computational investment yielded no throughput benefit—it achieved equivalent or slightly worse performance than both greedy algorithms (Table 4). This can be attributed to the high initial packing density: 75% of builds exceeded 70% bounding-box utilization after greedy initialization, leaving minimal room for metaheuristic improvement. The combinatorial search space also grows explosively with part count; with 30–45 parts per build, a 50-iteration LNS budget samples a negligible fraction of possible configurations.

For Polygon, the 13-second overhead is justified by the 13.5% reduction in plate count. In production environments where plate changes incur operator time and failure risk, this tradeoff favors Polygon despite its slower computation. As shown in Figure 3, Polygon Projection reduces plate count while maintaining acceptable compute time.

Qualitative Comparison

To evaluate practical impact, we compared Polygon Projection against PrusaSlicer's auto-arrange function, which uses libnest2d (Mészáros 2019)—an open-source C++ library implementing No-Fit Polygon (NFP) placement with First-Fit Decreasing selection and NLopt Subplex local optimization. Both algorithms were configured identically: 256×256 mm packing area, 6mm inter-part spacing, no border safety margin, and an 18×28mm exclusion zone for the Bambu X1C filament cutter. PrusaSlicer's arrange function exposes multiple quality settings. We compared against the default configuration: no part rotations and "Fast" geometry handling, which uses convex hull approximations rather than exact part boundaries. Our Polygon Projection method uses rasterized geometry at 1mm resolution and evaluates 90° rotation increments, representing a higher-fidelity but still computationally inexpensive approach. Notably, PrusaSlicer does not sort parts by height—libnest2d uses area-based ordering as is standard for 2D bin packing—whereas our method employs height-based sorting to minimize print time. Both Polygon Projection and Prusa placements are evaluated using the identical analytical cost model, ensuring differences reflect placement quality rather than slicer heuristics.

We generated $N = 333$ builds following the same procedure as Phases 1–2. Four builds were excluded where Prusa produced anomalous results ($R < 0.5$), likely due to edge-case geometry handling, yielding $N = 329$ for analysis.

Table 2 summarizes the comparison. Polygon Projection achieved a geometric mean throughput improvement of 5.7% (95% CI $[4.7\%, 7.3\%]$, Wilcoxon $p < 0.001$). Log-transformed throughput ratios showed mild non-normality (Shapiro-Wilk $W = 0.51$ for differences), reflecting occasional large wins by Polygon Projection; Wilcoxon signed-rank confirmation ensures robustness.

In practical terms, this improvement translates to a median savings of 19.5 minutes per build. For plate count, Polygon Projection won 81 builds (24.6%) versus 29 for Prusa (8.8%), with 219 ties. While Table 2 shows a higher mean plate count for Polygon Projection (3.42 vs. 3.23), the median is identical (3 plates) and the IQR for both methods is 1.5 plates. The elevated mean is driven by a single outlier where Polygon required 41 plates for a geometrically complex build that Prusa handled in 9 plates, suggesting sensitivity to edge-case geometries that warrants further investigation.

Compute times were comparable: Polygon Projection completed in a median of 3.0 seconds, while PrusaSlicer's arrange completed in approximately 3–10 seconds. The Prusa timing is approximate, as data collection via GUI automation precluded precise measurement.

Algorithm	R (mean $\pm$ std)	Median Plates	Median Time (s)	N
Bounding Box	$2.82 \times \pm 1.69$	8	<1	148
Polygon Projection	$2.79 \times \pm 1.47$	**7**	13	148
LNS	$2.66 \times \pm 1.41$	9	71	53

Table 4: Phase 2 results: Algorithm comparison. R = throughput ratio, d = Cohen's d vs. next-best algorithm.

Limitations

The model abstracts slicer internals. The scaling factors (s_v, s_Q, c_V) and the intra-island travel term $L_{\text{intra}}(\cdot)$ require calibration using slicer-reported summaries or empirical profiling. The NN-Tour proxy is fast but not guaranteed to match any specific slicer's global island ordering. The framework also assumes consistent process settings across parts; heterogeneous materials or per-part parameter changes may require extending the time model. Our Phase 3 comparison used PrusaSlicer's default auto-arrange configuration: "Fast" geometry mode with rotations disabled. Enabling full optimization (accurate geometry handling and rotation search) may reduce or eliminate the throughput advantage observed for our Polygon Projection method. Additionally, due to the challenges of GUI automation, we were unable to record precise wall-clock timing of Prusa's auto-arrange, and instead report approximate compute times in our Phase 3 analysis.

Conclusion

We presented a time-aware packing formulation that optimizes 3D printing throughput by coupling geometric placement with slicing-aware print-time estimation. Our cost model achieves strong correlation with slicer-reported times ($r = 0.90$, $\rho = 0.96$) without invoking toolpath generation, enabling rapid batch optimization.

A key empirical finding is that simple greedy packing suffices for this problem: polygon-based placement beat metaheuristic search at a fraction of the compute cost, as we hypothesize that high initial packing density leaves minimal room for iterative refinement. This is a useful negative result for practitioners considering more expensive optimization approaches. Against PrusaSlicer's production auto-arrange, our approach achieved 5.7% throughput improvement. We note that this comparison used Prusa's "Fast" geometry mode with rotations disabled (the default setting); enabling full optimization may reduce this gap, which we leave to future work. We also demonstrate that height-based part sorting may marginally improve throughput. Since PrusaSlicer's libnest2d arranges by area (standard for 2D bin packing) rather than height, incorporating height-aware ordering into commercial slicers represents a low-cost improvement opportunity at no computational cost.

Future directions include evaluating against commercial arrangers with full rotation optimization, exploring finer rasterization for tighter polygon interlocking, and extending the framework to multi-printer and multi-material scheduling.

Acknowledgements

This material is based upon work supported by the Office of Naval Research under Grant No. N00014-22-1-2714. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Office of Naval Research or the Department of Defense.

References

Jylänki, J. 2010. A Thousand Ways to Pack the Bin: A Practical Approach to Two-Dimensional Rectangle Bin Packing. In *Unrefereed Survey*.

Liu, W.; Chen, L.; Mai, G.; and Song, L. 2020. Toolpath planning for additive manufacturing using sliced model decomposition and metaheuristic algorithms. *Advances in Engineering Software*, 149: 102906.

Mészáros, T. 2019. libnest2d: 2D Irregular Bin Packaging and Nesting Library. C++ library implementing No-Fit Polygon placement with First-Fit selection.

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

Prusa Research. 2024. PrusaSlicer: G-code Generator for 3D Printers. Open-source slicer software, fork of Slic3r.

secnot. 2017. rectpack: Rectangle Packing Library. Python library implementing MaxRects and Skyline bin packing algorithms.

Zhou, Q.; and Jacobson, A. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797*.