Scalable Optimization Methods for Integrated Nesting and Scheduling in Additive Manufacturing

P. J. Nascimento, S. Moniz

University of Coimbra, CEMMPRE – ARISE, Department of Mechanical Engineering, Coimbra, Portugal

pnascimento@dem.uc.pt

Abstract *— Additive Manufacturing (AM) poses significant production scheduling challenges because scheduling and nesting decisions must be solved jointly. This work addresses the AM scheduling problem (AMSP), focusing on the nesting component, which remains the main computational bottleneck of existing approaches. Current nesting methods face scalability limitations, require costly preprocessing, and handle part rotations inefficiently. Building on a review of geometric tools for non-overlapping constraints, we propose a nesting framework that leverages rasterization, the Convolution Theorem, the Fast Fourier Transform (FFT), and modern graphics processing units (GPUs) to determine feasible part placements. Experiments show that finer rasterization resolutions and a predefined number of rotations do not necessarily increase run times, underscoring the framework's scalability. The framework is then embedded into two Biased Random-Key Genetic Algorithms (BRKGAs) to solve the AMSP. Preliminary results on benchmark instances show the BRKGAs are competitive with a state-of-the-art approach.*

Index Terms *— Nesting, Rasterization, Fourier transforms, Additive manufacturing, Scheduling.*

I. Introduction

The Additive Manufacturing (AM) market is growing rapidly, with a projected compound annual growth rate of about 23% through 2030 [1]. Its tool-free and layer-by-layer process keeps efficiency largely independent of part variety, making it well suited to low-volume and highly customized production. However, AM still suffers from slow production rates and high fixed operating costs, meaning machines must be used as efficiently as possible. As such, production scheduling gains particular relevance since good scheduling decisions are a path to maximize efficiency and overall productivity.

In AM, machines operate in parallel and a set of parts can be produced simultaneously, i.e. in a batch. Satisfying one-dimensional knapsack (capacity) constraints, however, does not guarantee that all parts in a batch can be physically arranged on the build platform. To achieve this, a 2D or 3D irregular packing problem must also be solved [2] alongside the scheduling problem. We refer to this integrated problem as the AM scheduling problem (AMSP), and since most AM technologies do not efficiently support vertical stacking, we consider a 2D irregular packing problem (i.e., the nesting problem) in which the projections of parts onto the build platform must not overlap.

Recent AMSP approaches that handle nesting without major simplifications [2–5] still share three main limitations. First, discretization-based nesting often lacks scalability since finer resolutions or larger build platforms substantially increase run time. Second, several methods require complex and computationally intensive preprocessing processes whose cost is far from negligible in the AMSP. Third, part rotations are frequently neglected.

This work addresses these limitations by proposing a rasterization-based nesting framework supported by the FFT algorithm and GPU acceleration, and by integrating it into two BRKGAs to solve the AMSP.

II. Geometric Tools for Nesting

Let $\mathbf{P}_j(\theta_j)$ denote the projection of part j rotated by θ_j, and $\mathbf{A}$ the placement area. Nesting requires that every part lie within $\mathbf{A}$ and that parts do not overlap. Four geometric tools are typically used to enforce these constraints: direct trigonometry, phi-functions, the no-fit polygon (NFP), and rasterization [6]. For direct trigonometry and phi-functions, they quickly become computationally intractable for complex shapes. The NFP and rasterization dominate large-scale heuristic approaches [7, 8].

NFP-based methods can be highly accurate and compact, but computing NFPs is costly for intricate parts and the cost of finding valid positions grows with the number of parts already placed [9]. Rasterization, in which parts and platform are represented as matrices, typically binary, is simpler to implement and, crucially, maps naturally onto modern GPUs widely used in rendering and vision [10, 11]. Our experiments below indicate that, with efficient algorithmic design and GPU support, rasterization is a compelling foundation for nesting within the AMSP.

III. The Nesting Framework

Let M be the $L \times W$ binary matrix of the placement area and G the $l \times w$ matrix of a part. Feasible placements correspond to positions where the cross-correlation Y between G and M is zero. The naive search has complexity $O\big((L-l+1)(W-w+1)lw\big)$. Instead, we exploit the Convolution Theorem: a convolution in the spatial domain is a point-wise product in the frequency domain. Using a zero-padded flipped part matrix $G_{\text{pad}}^{\text{flip}}$ matched to the size of M, the same-size circular correlation Y is obtained as

$$O = \mathscr{F}^{-1}\big(\mathscr{F}(M) \odot \mathscr{F}(G_{\text{pad}}^{\text{flip}})\big), \tag{1}$$

where $\mathscr{F}$ is the FFT, with overall complexity $O(k \log k)$ and $k = L \cdot W$. Cells of Y inside the region of interest equal to zero are feasible placement positions; positive values indicate overlap (Algorithm FINDPLACEMENT). The FFT-based cost

is independent of part size and shape, and is dramatically accelerated on GPUs.

A. Scalability Running FINDPLACEMENT 1 000 times each over seven platform sizes (up to 600×600, covering current AM machines at a 1 mm resolution) on an NVIDIA A100 GPU (CuPy) shows that run time is essentially insensitive to platform size and discretization, and that the small standard deviation confirms independence from part shape and platform occupancy. Moreover, because matrices of equal size can be stacked into a higher-dimensional array, the R rotations of a part can be evaluated in a *single* FFT call (G_{pad} of size $R \times 600 \times 600$), so considering a bounded set of rotations does not penalize run time. Both findings directly counter the scalability and rotation limitations of prior methods.

IV. The AMSP and BRKGA Approaches

A set of parts $\mathbf{J} = \{1, \ldots, n_p\}$ must be grouped into batches and assigned to non-identical AM machines $\mathbf{M} = \{1, \ldots, n_m\}$ to minimize the makespan. Vertical stacking is not allowed, so each batch must form a feasible nesting on the platform. Selective Laser Melting (SLM) is modeled, where the processing time of a batch combines a recoating term (driven by the tallest part), a scanning term (driven by part and support-structure volumes), and a setup term [12].

We embed the nesting framework into two BRKGAs [13]. A chromosome has $2n_p$ real-valued keys: the first n_p assign parts to machines and the rest define the placement sequence. The *standard* BRKGA (sBRKGA) follows the classical elite/mutant/biased-crossover scheme. The *enhanced* BRKGA (eBRKGA) adds two evolving keys and self-adaptive elite/mutant proportions, together with reset/shake perturbations triggered after a number of non-improving generations, to escape local optima [14, 15].

A placement policy (fitness evaluation) decodes each chromosome and packs parts sequentially across open batches, trying all rotations within batches where the platform area is not exceeded. To avoid invoking the FFT search unnecessarily, the vacancy- and density-vector pre-checks of [4] are applied before each placement attempt. Infeasible part–machine combinations are avoided by restricting the assignment keys to feasible ranges.

V. Preliminary Results

We use the benchmark of [4]: 100 part models with support structures and four SLM machines, organized into five classes (C1–C5) of increasing size (25 to 150 parts). The build platform and parts were discretized at 1 mm which took 0.07 s for all 100 different parts. Because GPUs were unavailable for these runs, experiments were performed on an Apple M1 Pro CPU (PyTorch), meaning the implementation is not fully optimized and results are therefore preliminary. Each instance was solved 10 times and averaged per class, and compared with the VNS of [4].

As Table 1 shows, both BRKGAs match or improve VNS solution quality, with the gap widening for larger classes at about 8–8.5% better makespan on C5. Run times to reach the best solution remain of comparable magnitude for C1–C3; for C4–C5 the BRKGAs need more generations but keep improving, whereas the VNS plateaus early. Since the BRKGAs are designed for GPU execution while the VNS suits CPUs, this comparison targets orders of magnitude rather than exact times. The eBRKGA is advantageous for smaller instances, where the sBRKGA can stall in local optima, while the sBRKGA is competitive and cheaper on larger ones.

Table 1. Preliminary results: average makespan (Sol, in hours), time to best solution (Best, s) and total run time (Total, s). VNS values from [4].

	sBRKGA			eBRKGA			VNS		
Cl.	Sol	Best	Total	Sol	Best	Total	Sol	Best	Total
C1	36.16	118	691	**36.05**	522	1127	36.16	767	1508
C2	58.08	1136	2117	**57.83**	2774	3792	58.94	1507	2354
C3	103.67	3001	4483	**103.01**	3996	5269	105.28	2293	2972
C4	61.60	5887	8880	**60.77**	6667	10342	64.32	1587	2556
C5	**81.23**	16620	17638	81.60	17279	22724	88.76	1734	2816

VI. Conclusions and Future Work

We characterized geometric tools for nesting and identified rasterization, combined with the FFT and GPU acceleration, as a scalable foundation for solving nesting within the AMSP. Experiments confirm that finer discretization and bounded rotations do not necessarily increase run time. Integrated into two BRKGAs, the framework yields preliminary results competitive with the state of the art even without a GPU-optimized implementation. Future work targets GPU-accelerated, fully optimized BRKGAs, larger and more complex AMSP instances, and 3D irregular-packing extensions, illustrating how powerful hardware can be harnessed for combinatorial optimization.

References

[1] Grand View Research, "Additive Manufacturing Market Size, Share & Trends Analysis Report, 2024–2030," 2024.

[2] J. Zhang, X. Yao, and Y. Li, "Improved evolutionary algorithm for parallel batch processing machine scheduling in additive manufacturing," *Int. J. Prod. Res.*, 2020.

[3] P. J. Nascimento, C. Silva, S. Mueller, and S. Moniz, "Nesting and scheduling for additive manufacturing: an approach considering order due dates," in *Operational Research (IO 2021)*, Springer, 2023, pp. 117–128.

[4] Z. Lu, K. Hu, and T. S. Ng, "Improving additive manufacturing production planning: a sub-second pixel-based packing algorithm," *Comput. Ind. Eng.*, vol. 181, 2023.

[5] P. J. Nascimento, C. Silva, C. H. Antunes, and S. Moniz, "Optimal decomposition approach for solving large nesting and scheduling problems of additive manufacturing systems," *Eur. J. Oper. Res.*, vol. 317, no. 1, pp. 92–110, 2024.

[6] A. A. S. Leao, F. M. B. Toledo, J. F. Oliveira, M. A. Carravilla, and R. Alvarez-Valdés, "Irregular packing problems: a review of mathematical models," *Eur. J. Oper. Res.*, vol. 282, no. 3, pp. 803–822, 2020.

[7] N. Chernov, Y. Stoyan, and T. Romanova, "Mathematical model and efficient algorithms for object packing problem," *Comput. Geom. Theory Appl.*, vol. 43, no. 5, pp. 535–553, 2010.

[8] J. A. Bennell and J. F. Oliveira, "The geometry of nesting problems: a tutorial," *Eur. J. Oper. Res.*, vol. 184, no. 2, pp. 397–415, 2008.

[9] E. K. Burke, R. S. R. Hellier, G. Kendall, and G. Whitwell, "Complete and robust no-fit polygon generation for the irregular stock cutting problem," *Eur. J. Oper. Res.*, vol. 179, no. 1, pp. 27–49, 2007.

[10] M. Schütz, B. Kerbl, and M. Wimmer, "Software rasterization of 2 billion points in real time," *Proc. ACM Comput. Graph. Interact. Tech.*, vol. 5, no. 3, 2022.

[11] L. Denis, R. Royen, Q. Bolsée, N. Vercheval, A. Pižurica, and A. Munteanu, "GPU rasterization-based 3D LiDAR simulation for deep learning," *Sensors*, vol. 23, no. 19, 2023.

[12] I. Kucukkoc, "MILP models to minimise makespan in additive manufacturing machine scheduling problems," *Comput. Oper. Res.*, vol. 105, pp. 58–67, 2019.

[13] J. F. Gonçalves and M. G. C. Resende, "Biased random-key genetic algorithms for combinatorial optimization," *J. Heuristics*, vol. 17, no. 5, pp. 487–525, 2011.

[14] A. A. Chaves, J. F. Gonçalves, and L. A. N. Lorena, "Adaptive biased random-key genetic algorithm with local search for the capacitated centered clustering problem," *Comput. Ind. Eng.*, vol. 124, pp. 331–346, 2018.

[15] C. E. Andrade, T. Silva, and L. S. Pessoa, "Minimizing flowtime in a flowshop scheduling problem with a biased random-key genetic algorithm," *Expert Syst. Appl.*, vol. 128, pp. 67–80, 2019.