ASTES

Advances in Science, Technology and Engineering Systems Journal
Vol. 10, No. 5, 51-65 (2025)
www.astesj.com
Special Issue on Innovation in Computing, Engineering Science & Technology

ASTESJ
ISSN: 2415-6698

Optimization of Sheet Material Layout in Industrial Production Using Genetic Algorithms

Chiang Ling Feng*

Yu Da University of Science and Technology, Bachelor Degree Program of IOT Engineering and Applications, Miaoli County, 361, Taiwan

A R T I C L E I N F O

Article history:
Received: 25 August, 2025
Revised: 11 October, 2025
Accepted: 13 October, 2025
Online: 25 October, 2025

Keywords:
Genetic Algorithm
Fitness Function
Optimization
Mutation
Sheet Material Similarity

A B S T R A C T

We address irregular polygon nesting on sheet materials with a lightweight evolutionary framework that operates directly in the layout space. The method formalizes multi-term fitness combining utilization, overlap penalties, spacing regularity, and local alignment, with all components normalized before aggregation. Feasibility is enforced by an AABB–SAT pipeline and validated via analytic ground-truth cases, degenerate contacts, and cross-implementation checks. The evolutionary core encodes absolute positions and orientations, uses geometric crossover and local repair, and is assessed under repeated runs with confidence bands. A systematic sensitivity study over mutation rates and population sizes reveals a stable operating regime that balances exploration and convergence and consistently yields low waste ratios on controlled synthetic benchmarks. We discuss limitations (e.g., late-stage densification) and outline hybridization paths with stronger geometric kernels. The framework thus provides a reproducible and engineerable baseline that can be integrated into industrial nesting workflows and extended to real production datasets.

1. Introduction

Sheet material layout has wide-ranging applications in both everyday life and industrial sectors. In furniture manufacturing, materials such as wood, synthetic boards, and metal sheets need to be optimized to minimize material waste and ensure production efficiency [1]. In construction and renovation projects, large sheets such as gypsum boards, laminates, and wood panels need to be arranged to cover surfaces like walls, ceilings, and floors efficiently, reducing material waste and costs. In the packaging industry, optimizing the layout of cardboard for boxes, packaging materials, and containers can significantly reduce packaging costs. In the textile industry, arranging fabrics of different sizes and shapes to minimize waste and increase yield is crucial for apparel manufacturing, home textiles, and industrial textiles. In the printing industry, arranging printed sheets, labels, and stickers can save paper and ink, enhancing printing efficiency. In metal processing, arranging metal sheets for cutting, stamping, welding, and machining is a common operation; efficient layouts reduce material waste and improve production efficiency. In food processing, arranging food products to fit packaging containers, baking trays, or grills ensures efficient production and reduces food waste. In the manufacture of paper products like envelopes, cards, books, and boxes, efficient layout of paper and cardboard is essential to minimize waste. In aerospace engineering, material layout and cutting optimization are critical for manufacturing aircraft and spacecraft components, reducing costs and improving performance. In electronics manufacturing, closely arranging circuit boards and semiconductor wafers minimize waste and enhance production efficiency. These are everyday applications of sheet material layout in various fields, demonstrating how efficient material usage can lower costs, reduce resource waste, and improve production efficiency.

In the field of irregular nesting and cutting/packing, the earliest application of evolutionary search to polygon nesting was by Jakobs. His motivation was highly industrial: in real-world manufacturing, part geometries are irregular, rotations are continuous, and multiple interactions occur simultaneously. Traditional rule-based cutting methods quickly ran into combinatorial explosions. His core approach was to use a genetic algorithm (GA) to encode the placement order and orientation of each piece as chromosomes, relying on crossover and mutation to escape local optima; geometric feasibility was maintained by simple "line-segment collision tests." The weakness was that the geometric handling was crude—holes and concave polygons often caused misjudgments—and the GA's convergence speed and parameter sensitivity were high, requiring extremely long runtimes to produce acceptable layouts. These limitations were already visible in his 1996 experiments [1]. While this line of work encodes sequences and relies on a geometric checker, our

*Corresponding Author: Chiang Ling Feng, No. 168, Hsueh-fu Rd., Tanwen Village, Chaochiao Township, Miaoli County, 361 Taiwan, (R.O.C), +886 921637358, acclaim0629v@gmail.com

method shifts the decision space from sequence to direct layout, avoiding a heavy geometric decoder and enabling immediate feasibility probing in the layout plane.

Building on the "first determine order, then place" paradigm, [2] transplanted the rectangular Bottom-Left (BL) philosophy to polygonal nesting. Their motivation was to use a minimal rule set to gain speed and stability, enabling frequent reuse within higher-level searches such as GA or SA. The principle is to place each subsequent piece at the lowest and leftmost feasible position, combined with simplified geometric tests such as boundary scanning. The fatal weakness of BL lies in its extreme sensitivity to sequence and its tendency to create "floating voids," causing significant waste in high-density packings. Consequently, later works typically use BL only as a construction heuristic rather than a final optimizer. The BL family's sequence sensitivity motivates our direct-layout evolution that explores absolute positions/angles with on-the-fly feasibility checks, rather than committing to a single greedy frontier per sequence.

To make placement more reliable, [3] greatly improved the computability of the No-Fit Polygon (NFP). Their motivation was to transform feasibility detection from "sampling-based probing" into "geometric determinism." Conceptually, an NFP is formed by sliding one polygon around another, tracing the locus of all positions where the two polygons are just touching; any point outside this NFP represents a non-overlapping feasible position. Early NFP algorithms, however, often failed with concave, curved, or holed shapes due to numerical instability, and the generation cost increased significantly with many parts. Instead of investing upfront in full NFP generation, we trade deterministic boundaries for stochastic feasibility sampling, which is lightweight and portable at prototyping time, and can later be hybridized with NFP if tighter density is required.

In 2006, [4] brought the BL approach to its highest level. Their motivation was to evolve BL from a "fast but coarse" heuristic into an algorithm capable of handling curved edges and holes, while outperforming previous methods on industrial benchmarks. The core principle remained a constructive bottom-left-fill, but with more efficient geometric state management and robust collision detection, allowing it to function under diverse geometric representations. The limitation, however, was intrinsic: no matter how strong a constructive heuristic is, it remains sequence-dependent, and for ultra-dense or large-scale rearrangements, solution quality still lags behind powerful combinatorial or hybrid metaheuristic methods. Because constructive kernels remain sequence-bound, we operate directly in layout space and treat rotation as a first-class, continuous variable, reducing reliance on sequence heuristics during early exploration.

To fully resolve NFP instability, [5] proposed a complete and robust NFP generation algorithm (the "orbital/sliding" method), systematically addressing all prior failure cases and making NFP a true industrial geometric backbone. The motivation was to "engineer" what had previously been geometric black magic. The method systematically tracks the contact trajectory between polygons without global discretization. Its main drawback remains computational cost: NFP generation is not itself a solver—no matter how precise it is, it must still be embedded within an outer optimization process for efficiency. We defer heavy geometric cores by using AABB/SAT checks plus angle sweeps; this yields a zero-infrastructure path to usable layouts and preserves the option to "plug in" NFP later.

The same team later (2009) integrated Simulated Annealing (SA) into Best-Fit-type rectangular placement. Their motivation was to use temperature-controlled stochastic perturbations to escape congested suboptimal sequences; the principle was to adjust part order and orientation through SA, then apply an efficient constructive placement. The limitation was that classic SA required empirical tuning of parameters and cooling schedules, and the computational time remained high for large instances. Nevertheless, this hybrid line of research directly influenced many subsequent industrial systems that combine a metaheuristic outer loop with a fast constructive placement kernel [6]. Rather than optimizing sequences than decoding, we evolve layouts in place, so each perturbation has immediate geometric meaning without a separate decoding stage.

In [7] and [8], the authors published two tutorial-style surveys that organized geometric primitives, overlap detection, data structures, and industrial benchmark datasets. Their motivation was to establish a shared terminology and reproducible baseline for comparison. The core principles reviewed include NFP, potential field/distance field, scan-line, and raster-based representations. Their shortcoming was not methodological but infrastructural: at that time, there were no unified open benchmarks or reproducible codebases, leading to friction in cross-study comparisons. We complement this toolbox by demonstrating that a minimal geometry stack can still produce stable convergence with repeated-run statistics, forming an accessible baseline for rapid experimentation.

To overcome the "constructive deadlock" problem, [9] introduced Beam Search. Their motivation was to retain multiple parallel solution branches to avoid getting trapped by greedy heuristics. The method preserves a limited number of best partial sequences (beam width) at each level, guided by NFP-based geometric scoring. The drawback is that memory and evaluation cost grow rapidly with beam width, requiring careful pruning and heuristic ranking to scale effectively. Instead of widening the sequence beam, we search over layouts directly, which naturally supports interactive guidance and constraint injections (keep-out zones, clearances) at feasibility level.

In the "strong local improvement" direction, [10] applied Tabu Search (TS) to two-dimensional non-guillotine cutting. The motivation was to exploit TS's memory mechanism to prevent cycling around similar sequences or orientations. The principle involves iterative neighborhood operations (swaps, rotations) combined with a tabu list and refined feasibility checks. The weakness is that performance depends heavily on neighborhood design and weight parameters, and geometric evaluation can be computationally expensive per step. We avoid expensive neighborhood evaluations on a decoder by making feasibility checks cheap and local (bounding boxes or SAT), which keeps per-step cost predictable under continuous rotation.

In [11] and [12], the authors demonstrated the general applicability of Simulated Annealing to two-dimensional (including non-guillotine) cutting problems. Their motivation was to use a unified framework to handle various industrial types. The principle treats overlapping area or utilization as an energy function and uses temperature-driven stochastic wandering to seek improvement. Limitations include slow convergence and high sensitivity to parameter tuning; without strong geometric modules like NFP, both solution quality and speed suffer. We keep the stochastic spirit but move randomness to the physical layout space with stepped angle sweeps, improving observability and designer-in-the-loop steerability.

More recently, [13] introduced a semi-discrete representation into nesting. Their motivation was to reduce continuous geometric collision detection to one-dimensional segment interactions, allowing BL-fill-type constructive algorithms to run in milliseconds, making them viable high-frequency cores for outer metaheuristics. The principle converts both parts and sheet geometry into vertical segment sets using scan-lines, ensuring that "non-overlapping segments ⇒ non-overlapping shapes." The trade-off lies in geometric fidelity: with coarse resolution, highly concave or narrow shapes may be overly constrained, leading to conservative "blank zones" and reduced utilization. Semi-discrete acceleration is orthogonal to our idea; in future we can use semi-discrete frontiers as candidate generators while retaining our direct-layout evolution as the outer search.

From an evolutionary computation perspective, [14] and [15] became common industrial approaches. Their motivation was to reduce infeasible chromosomes by using random-key encoding, ensuring that ordering and rotations naturally fall within feasible domains. The principle maps random keys into sequences, then evaluates them with BL/NFP placement cores. The limitation is that if the placement kernel is only BL-based, the overall method remains order-dependent, and the GA often requires expensive local improvement or hybrid mathematical programming to squeeze out the last few percentage points of utilization. To avoid the genotype–geometry semantic gap (similar keys, very different layouts), we manipulate the layout itself; every mutation immediately changes geometry, not just the decoding order.

Among swarm-based heuristics, [16] decomposed Particle Swarm Optimization (PSO) into "sequence optimization + geometric placement." Their motivation was to test PSO's viability on complex, irregular, open-dimension strip problems. The principle represents each particle as a sequence and orientation vector, updated through global and local bests, sometimes combined with local search. The main weakness lies in the sensitivity of inertia and weight parameters, and if the placement subroutine is weak, particle evaluations become extremely noisy. A similar approach appears in [17], who used PSO's structural simplicity to explore sequence space quickly. The principle remained the same—feasibility and scoring delegated to placement modules (typically BL or raster-based)—and used population perturbation to avoid premature convergence. Its limitation is inherited from constructive placements: under tight-fitting or multi-angle conditions, large unusable voids persist. We adopt the "sequence-placement split" spirit but collapse them into a single layout-space search, reducing reliance on a fragile placement subroutine.

In [18]'s rubber-band–based visual nesting system, the motivation is thoroughly engineering-driven: it enables engineers without an optimization background to literally "see" geometric tightening and voids. The basic idea is to simulate virtual rubber bands stretched around part boundaries, coupled with semi-automatic human–computer interaction to adjust the layout on the fly. Its weakness lies in its role as decision support rather than a path to fully automated near-optimal layouts; to reach optimal or near-optimal performance, it still needs to be coupled with a formal optimizer. From a broader perspective, the survey by [19] on mathematical models aims to clarify, in a single treatment, the strengths, weaknesses, and scalability of ILP, nonlinear, and constraint-programming formulations. The central message is that rigorous models provide strong bounds or even optimal solutions on small to medium instances, but at industrial scale they typically need to be hybridized with heuristics or decomposition, with the chief limitation precisely in scalability and the costs of geometric linearization. We position our framework as a bridge: quick, visual, constraint-friendly layouts without geometry heavy-lifting, and compatible with formal optimizers for downstream tightening.

The method we propose intentionally pursues a path quite different from the mainstream "sequence plus geometric decoder" lineage. It operates directly in the physical coordinate plane by randomly sampling drop locations and angles, admitting layouts whenever they pass a very simple feasibility check, and then applying an ultra-lightweight, GA-like wrapper to make in-place micro-adjustments in layout space. Compared with canonical approaches centered on NFP, BLF, or Beam Search, the defining feature is that the decision variables are not "piece sequences and poses," which are easy to decode and recombine, but the concrete set of "absolute coordinates and angles of pieces already placed." As a result, what would normally be entrusted to a professional placer to delineate—namely the geometric feasibility region—is here replaced by large numbers of random trials and a minimal axis-aligned bounding-box overlap test that effectively gambles for feasibility. This direct wandering in layout space is attractive because implementation cost is extremely low, module boundaries are thin, and almost no geometric infrastructure is needed to get something "up and running." For rapid proof-of-concepts, coarse estimation of attainable packing levels for a given mix of parts, or quick incorporation of additional engineering constraints such as safety clearances and boundary offsets, it is indeed handy. More importantly, it treats rotation—a continuous degree of freedom that explodes complexity in traditional NFP/BLF—as a stepped angle sweep with local randomized retries, avoiding intricate continuous collision computations and keeping maintenance comparatively light. The method is also distinctive in its objective design: it proxies utilization with total used area and treats residual micro-voids as homogeneous waste. This is acceptable in early exploration; however, relative to common NFP-plus-scorer schemes that emphasize "boundary adhesion," "void connectivity," or BLF's "frontier smoothness," our objective is overly macro. It deprives the search of local signals such as "one more small nudge unlocks a large interlock gain," making it difficult for evolution to climb from loose feasibility to tight, high-density packing. Traditional methods often insert local reordering or small-angle neighborhoods to smooth the energy landscape and guide convergence toward denser states. In addition, evolving directly in layout space makes it difficult to reuse geometric work: in the NFP/BLF world, a new sequence triggers immediate, cacheable, fast re-placement; in our approach, a slight perturbation to a random drop point almost requires refining feasibility from scratch, which reduces computational reuse and becomes a performance bottleneck at scale. Even so, in an engineering context, this "zero-infrastructure, instantly runnable" framework still has a practical niche. When the goal is to quickly estimate attainable utilization across several part-mix scenarios, or to generate a batch of rough layouts as starting points for downstream refinement—say, passing them to a small NFP placer or to an interactive rubber-band system—it delivers "useful though not optimal" drafts with very low development friction. It also accepts engineering constraint minimum clearances, keep-out zones, or simple multi-stock logic—without spinning up heavy geometry libraries and complex data structures. If we intend to push further, a natural route is to preserve the "direct layout" façade while abandoning pure randomness in placement: use a BLF-style frontier as a candidate-point generator, apply a coarse grid or semi-discrete scan to prune clearly infeasible regions, and then execute a small continuous adjustment. At the same time, redefine the "genome" as "placement order plus angle keys," so that mutation and crossover trigger re-placement and return evolution to a meaningful semantic space. As for geometric feasibility, even without full NFP, replacing bounding boxes with the Separating Axis Test (SAT) or a simplified convex-decomposition check will make the feasible region closer to reality and thus extract more value per unit of computation. In short, our method trades "precision and extensibility in the theoretical sense" for "extreme simplicity and immediate operability," which indeed saves time during prototyping.

Across GA-based sequence optimizers, constructive BL/BLF/NFP decoders, and their hybrids (Beam, TS/SA), the prevailing pipeline "decide sequence → decode geometrically → evaluate" has delivered strong results but also structural dependence on a heavy geometric core. This architecture struggles with continuous rotation (requiring coarse discretization or large precomputation), incurs infrastructure and portability costs (robust NFP/BLF implementations are complex to develop and transfer across stacks), and often relies on non-reproducible industrial datasets, hindering comparative analysis. Furthermore, while visualization and human-in-the-loop systems exist, they are typically auxiliary and not tightly integrated with the search mechanism.

Our work addresses these gaps by changing the decision space: rather than optimizing sequences and invoking a decoder, we perform direct layout evolution on absolute positions and angles, with on-the-fly feasibility checks (AABB or SAT) and stepped angle sweeps that treat rotation as a first-class continuous degree of freedom. This yields a zero-infrastructure path for rapid prototyping, easy injection of engineering constraints (clearances, keep-out zones, material grain), and immediate visualization that supports designer-in-the-loop workflows. To ensure reproducibility, we provide a synthetic benchmark with repeated-run statistics (means, dispersion, confidence bands), and we outline hybridization paths: semi-discrete BLF frontiers as candidate generators and NFP modules for late-stage tightening. In short, our contribution is not a parameterization of a known GA; it is a re-factoring of the search space and feasibility construction, turning a decoder-bound pipeline into a lightweight, directly operable layout search that can later be fused with classical geometric cores when density targets demand it.

The innovation of our algorithm does not lie in introducing a brand-new theory. Rather, it reconstructs the problem-solving workflow for nesting from an aggressively simplified, practical-engineering angle, yielding system behavior fundamentally different from the literature in terms of implementation barrier, computational flexibility, rotational freedom, and human–computer fusion. Prior research—GA [1], bottom-left strategy [2], and robust NFP and BLF variants [5] and [6]—reduce the task to a sequential pipeline of "decide placement order → invoke a precise placer → evaluate utilization." While this design can approach optimal packing, it demands substantial geometric computation, data structures, and preprocessing. Our approach goes the other way: it throws shapes directly into the real coordinate system, replaces strict boundary tracking with random sampling and bounding-box checks, and substitutes dynamic testing for static modeling. Theoretical rigor is partially relinquished in exchange for high portability and instant responsiveness. The conceptual innovation is not to defeat every traditional algorithm, but to redefine what "good enough" means—not treating near-global optimality as the sole objective but centering the value proposition on "rapidly generating reasonable layouts that can be extended with additional constraints."

The first locus of innovation is a shift in problem modeling. Traditional nesting algorithms are built on geometric models, emphasizing exact tangencies, minimum bounding boxes, or NFP descriptions. We instead treat every vertex of a shape as a physical point and test feasibility using randomly generated coordinates, effectively merging discretization and geometric checking into a single layer of computation. This reframes the model from a theoretical problem to an engineering simulation problem. Relationships among shapes no longer rely on complex topological routines; rather, simple numerical probes create a condition of "practical non-overlap." The system therefore runs without heavyweight geometric modules such as CGAL or Shapely, which is especially suitable for early design or cross-language embedding. At a deeper level, part of the classical NP-hard subproblem—geometric feasibility—is randomized, with the feasible space approximated through the statistical distribution of many samples rather than being delineated deterministically. This Monte-Carlo-style geometric sampling can prove more stable than expected in cases with high rotational freedom and heterogeneous part sets.

The second innovation lies in the minimalist, flexible evolutionary wrapper. We do not adopt classical GA encodings, crossover schemes, selection, or mutation formulas. Each candidate solution corresponds directly to a concrete layout, and mutation is implemented as element swaps at the layout level. Although this may appear primitive, it shifts the focus of evolution from "abstract genes" back to "concrete layout behavior." Where [14] and [15] are theoretically elegant yet often suffer a semantic gap—similar genotypes yielding vastly different layouts—our scheme avoids that gap. Every mutation or local rearrangement directly affects the geometric end state. In other words, the evolutionary operators move from "exploring ordering space" to "exploring behavioral layout space." From a computational-intelligence perspective, this is a reinterpretation: it stops pursuing the formal aesthetics of genotypes and instead treats evolution as a physical experiment, with concrete positions, angles, and distances feeding back into fitness. While this reduces analytical tractability, it increases observability and steerability, making the method particularly fitting for designer-in-the-loop workflows, where humans can literally watch shapes move and understand the evolutionary process instead of peering into a black box of numbers.

A third innovation is the natural handling of continuous rotation. In the literature, NFP and BLF typically address rotation by discretizing angles—often every 15° or 30°—and precomputing NFPs at those angles, which is time- and memory-intensive. Our method randomizes rotations alongside positions, allowing angle to be an evolvable degree of freedom. Rather than fixing angle discretization a priori, it sweeps angles during placement and checks feasibility on demand. Although coarse, this avoids the complexity of precomputed NFPs and preserves a statistically continuous relationship between rotation and translation. Especially under industrial conditions with diverse shapes and tolerance for minor micro-overlaps, this "on-the-spot angle sweep" aligns better with actual processes than fixed discretization. Direct random search over continuous parameters yields greater flexibility in representing part posture without building a separate rotational subspace model.

A fourth innovation appears in extensibility and human–computer collaboration. The system is almost free of tightly coupled data structures; placement conditions, scoring functions, and boundary settings are separable modules. This makes it easy to integrate perceptual components such as vision or CAD annotations, external scorers such as cost functions or machining-sequence constraints, and to present every trial step visually in an interactive interface. Compared to industrial-grade BLF [4] and [5] or Beam Search [9], which relies on more complex algorithmic machinery, our approach functions as an open framework that slots naturally into GUIs. Designers can tune weights or guide directions during evolution. This co-creative mode reframes nesting from fully automated "black-box optimization" to "semi-automated co-evolution," emphasizing flexible adjustment and immediate visual feedback.

Finally, and most fundamentally, our work reframes a traditionally optimality-driven problem as an engineering adaptivity problem. We explicitly accept a strategy of "near optimal yet stable, fast, and visual," treating randomness and approximation as integral design choices, not defects. While this stance may appear less rigorous academically, it is highly innovative from a systems-engineering point of view. Real-world nesting rarely has a single objective; it balances time, material, visibility, safety margins, and machining order, among others. Our architecture embraces these heterogeneous conditions within an open probing framework, transforming nesting from a closed mathematical optimization exercise into a continually evolvable engineering simulation system. In summary, innovation is not a single technical breakthrough, but a paradigm shifts bridging theory and practice. It redefines both the problem space and the solving logic: from explicit geometric boundaries to statistical feasibility, from strict genotype design to behavioral evolution, from fixed angle discretization to continuous random exploration, and from closed algorithms to collaborative simulation platforms. Together, these layers form a "stochastic–approximate–operable" nesting mindset whose value lies not in squeezing out the highest possible utilization, but in achieving the most flexible solution architecture at the lowest cost. If coupled later with a traditional geometric core—such as semi-discrete BLF [13] or Beam Search [9]—the system can combine speed with high density, serving as a bridge between theoretical methods and industrial engineering.

We clarify that our use of a genetic algorithm is not textbook instantiation, but a domain-specific evolutionary core tailored to irregular nesting. Rather than optimizing a sequence that is later decoded by a geometric placer, the genotype represents absolute geometry: everyone is a complete layout encoded as the list of part centroids and orientations in the sheet coordinate system. This change of decision space is substantive, because it removes the reliance on a separate decoder and allows feasibility, rotation, and constraint handling to be treated as first-class, in-loop operations rather than external subroutines. The initialization reflects this direct-layout stance. Candidate layouts are constructed in the physical plane, with part locations proposed by stochastic sampling over the sheet and with angles treated as continuous variables that are explored through a stepped sweep during placement. To make early generations productive rather than purely random, we draw initial locations from a coarse spatial grid that respects the sheet bounds and safety clearances, and we optionally perturb a fast constructive draft when available. The objective is to seed the population with layouts that are already interpretable in geometric terms, so that variation operators act on meaningful spatial structures instead of abstract permutations.

Feasibility is enforced during evaluation by a two-stage geometric check. For every part placement we first use an axis-aligned bounding-box filter as a constant pretest to eliminate obvious collisions. Candidates that pass this screen undergo a separating-axis test on the current polygon representations; this avoids the numerical fragility of pure raster approximations and removes the need to precompute no-fit polygons at multiple angles. If a collision is detected, the evaluation does not immediately discard the individual. A local repair routine attempts a bounded sequence of micro-translations and micro-rotations around the offending part, guided by the contact normal revealed by the separating-axis test, to reestablish feasibility without altering the remainder of the layout. Only if these localized adjustments fail is an overlap penalty imposed; in this way feasibility and repair are embedded in the evolutionary loop and contribute directly to fitness.

The fitness function is likewise specific to the nesting task. We measure sheet utilization by the ratio of covered area to sheet area and combine it with penalties for unresolved overlaps and for violations of prescribed clearances and keep-out regions. These composite objectives reward dense packing while preserving engineering semantics that are often absent from generic formulations. Because rotation is continuous, we do not discretize it globally; instead, during evaluation each part's orientation is locally refined through a small angle sweep around its current value, which smooths the objective landscape and allows the repair routine to exploit nearby interlocking opportunities that rigid discretization would miss. Selection proceeds by rank with a modest elite fraction preserved to stabilize progress without collapsing diversity. The crossover operator is geometric rather than index based. Parents contribute spatially connected clusters of parts—contiguous with respect to adjacency in the current layout rather than contiguous in the chromosome order—and the child inherits these clusters as coherent blocks. After insertion, the repair routine reconciles local conflicts at the cluster boundaries. This operator respects interlocking substructures that matter for nesting, which single-point or two-point crossovers tend to destroy. Mutation acts at three granularities. Small Gaussian jitters perturb coordinates to fine-tune contacts; micro-angle adjustments explore orientations in the immediate neighborhood of a promising pose; and occasional pairwise displacement exchanges resolve local deadlocks by nudging two parts in opposing directions along their separating axes. Mutation rates are not fixed as global hyperparameters; they adapt to measured population diversity, which we compute as the average Euclidean distance across genomes in the combined position–angle space. When diversity collapses, mutation intensity rises to restore exploration; when diversity is healthy, the rate subsides to consolidate improvements.

Because the algorithm searches directly in layout space, termination and restart policies are designed to prevent stagnation in tight configurations. If the best fitness does not improve over a fixed window of generations, we trigger a light restart that injects new individuals while preserving a small elite set, thereby retaining useful substructures without reinitializing the entire population. This approach keeps the search mobile under high density while avoiding the cost of reconstructing geometric infrastructure. For reproducibility and to move beyond descriptive reporting, every parameter setting is evaluated across multiple independent runs with controlled random seeds. We summarize performance by the mean and standard deviation of the waste ratio and include confidence intervals and coefficients of variation to characterize dispersion. Where we compare structural choices—such as geometric versus index-based crossover, the presence or absence of the local repair routine, or the addition of the separating-axis test beyond bounding boxes—we treat these as ablation factors and assess their impact across repeated runs using non-parametric significance tests appropriate for stochastic optimizers. In this manner the empirical section demonstrates that the specialized operators materially affect feasibility rate, convergence speed, and final utilization, which would not be captured by a generic GA template.

We also report the computational profile in terms germane to nesting. The per-generation cost is dominated by collision checks and repairs; with bounding-box pretests most infeasible candidates are rejected in constant time, while separating-axis tests scale with the number of polygon edges involved. To manage this cost, we cache rotated vertex coordinates and edge normal for frequently used orientations encountered during micro-angle sweeps, and we evaluate individuals in parallel since feasibility checks are independent across genomes. These engineering choices, though mundane, are essential to making a decoder-free evolutionary search practical on realistic instances.

Framed in this way, the manuscript now makes clear that the contribution is not a recitation of textbook operators with tuned mutation rates or population sizes. The novelty lies in relocating the genotype from sequence space to the physical plane, embedding feasibility and repair as first-class evolutionary mechanisms, treating rotation as a continuous decision variable handled in situ rather than by heavy precomputation, and designing crossover and mutation to preserve and exploit spatially meaningful substructures. The resulting method provides a lightweight, decoder-free path to usable layouts that can stand alone in early design and can later be hybridized with classical geometric cores when still higher density is required.

The structure of this paper is as follows: Section 2 describes the basic principles of the genetic algorithm we adopted, along with some details before simulation. In Section 3, the algorithm is implemented in Python, and conclusions are drawn in Section 4..

2. Algorithm

In this study, the genetic algorithm (GA) is not treated as an abstract evolutionary model but as a dynamic learning process tailored for optimizing shape arrangement and material utilization. Each “individual” represents a specific sheet layout scheme, and its “chromosome” consists of multiple “genes,” where each gene corresponds to a single shape’s spatial coordinates, rotation angle, and material attributes. This encoding strategy enables the algorithm to handle multidimensional optimization involving both position and orientation, accurately reflecting the geometric relationships among shapes in real cutting scenarios. In each iteration, the fitness function is no longer a single metric based solely on material utilization rate but a composite evaluation function that integrates several factors, including material waste ratio, shape overlap penalties, local density distribution, and layout uniformity. This design gives the fitness function physical and geometric meaning, allowing it to more realistically represent the practical quality of a layout. To prevent the population from falling into local minima, the selection and crossover mechanisms adopt a multi-phase hybrid strategy: in the early generations, selection favors individuals with higher fitness to accelerate convergence, while in later stages, stochastic selection and partial retention of low-fitness individuals are introduced to maintain diversity and promote cross-regional exploration. During crossover, positional and angular data are exchanged using real-valued crossover operations instead of traditional binary encodings, enabling the generation of new geometric combinations in continuous space.

The mutation mechanism is spatially aware: when attractive or repulsive forces between shapes cause local crowding, mutation operations perform small displacements or rotational adjustments within feasible regions, effectively mimicking human fine-tuning behavior in manual layout design. Through this directional mutation strategy, the algorithm continually improves local structures without disrupting overall stability. The termination conditions are not limited to a fixed number of iterations but also include the stabilization of fitness change rates and the convergence of material waste area, ensuring that the final solution is both stable and practically viable. The basic process of a genetic algorithm is illustrated in Figure 1.

Overall, the process forms an intelligent evolutionary dynamic, in which the population of solutions self-adjusts, differentiates, and aggregates within the search space, gradually approaching the optimal layout under multiple constraints. In this way, the genetic algorithm transcends the boundaries of a theoretical model and becomes an industry-grade optimization strategy—capable of perceiving geometric interactions, performing adaptive learning, and exhibiting self-organizing behavior throughout the evolutionary process.

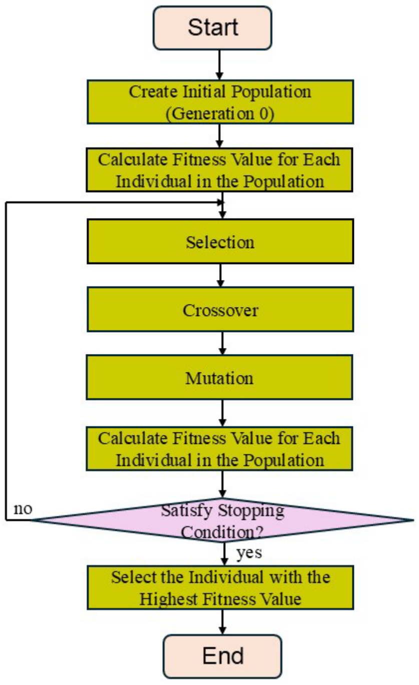
Start
Create Initial Population (Generation 0)
Calculate Fitness Value for Each Individual in the Population
Selection
Crossover
Mutation
Calculate Fitness Value for Each Individual in the Population
Satisfy Stopping Condition?
no
yes
Select the Individual with the Highest Fitness Value
End

Figure 1. Basic process of a genetic algorithm

2.1. *Fitness Function*

In this study, the fitness function is no longer limited to a simple evaluation of material utilization or waste ratio; instead, it is redefined as a multi-objective evaluation model with physical and geometric significance, capable of reflecting spatial interactions, layout coordination, and material efficiency simultaneously. Since the two-dimensional irregular sheet layout problem is a highly nonlinear, multi-variable combinatorial optimization problem with numerous local optima, traditional single-indicator approaches—such as the ratio of utilized to total area—fail to capture the complex relationships among shapes. Therefore, the proposed fitness function F integrates multiple factors, including *waste minimization*, *overlap penalty*, *spatial proximity*, and *alignment coherence*, through a weighted linear combination expressed as follows:

$$F = \omega_1 f_{util} - \omega_2 f_{overlap} - \omega_3 f_{dist} - \omega_4 f_{align} \tag{1}$$

Here, f_{util} represents the material utilization function, quantifying the ratio of effective used area to total sheet area; $f_{overlap}$ is the overlap penalty function, penalizing invalid overlapping between shapes; f_{dist} denotes the distance distribution function, measuring average spacing among shapes; and f_{align} represents the alignment coherence function, evaluating local edge alignment among neighboring shapes. The weights w_1, w_2, w_3, w_4 control the relative importance of these factors and are adaptively tuned based on practical application requirements. The material utilization function f_{util} is defined as:

$$f_{util} = \frac{A_{\text{used}}}{A_{\text{total}}} \tag{2}$$

where A_{used} is the effective covered area of the shapes and A_{total} is the total sheet area. The closer this value approaches 1, the more compact and efficient the layout is, indicating a higher-quality arrangement. The overlap penalty function f_{overlap} ensures that no physical intersection occurs among shapes, formulated as:

$$f_{overlap} = \sum_{i,j} \max(0, A_{ij}) \tag{3}$$

where A_{ij} represents the overlapping area between shapes i and j; if no overlap exists, the value is zero. This component acts as a constraint penalty, any overlap decreases overall fitness, forcing the algorithm to evolve toward feasible, non-overlapping layouts. The distance distribution function f_{dist} models the *attraction–repulsion behavior* between shapes. For each shape i, the distance d_{ij} to neighboring shapes $j$is computed, introducing a balancing parameter α:

$$f_{dist} = \frac{1}{N} \sum_{i,j} \left(\frac{1}{d_{ij}+\epsilon}\right)^{\alpha} \tag{4}$$

where N is the total number of shapes and ϵ is a small constant to avoid division by zero. When shapes are excessively crowded, this value increases and lowers fitness; when they are too far apart, material utilization decreases. The algorithm dynamically adjusts w_3 and $\alpha$to balance compactness and spacing. The alignment coherence function f_{align} evaluates the degree of alignment between neighboring shapes' edges. The angular difference θ_{ij} between adjacent shapes is computed, and only those within a distance threshold $D_{\text{th}} = 10.0$ are considered relevant. The function is expressed as:

$$f_{align} = \frac{1}{M} \sum_{i,j} \exp(-\beta \mid \theta_i - \theta_j \mid) \cdot \delta(d_{ij} < D_{\text{th}}) \tag{5}$$

where δ is an indicator function (1 if the condition holds, 0 otherwise), and β controls sensitivity to angular differences. This formulation ensures that alignment is evaluated only for nearby shapes, eliminating the influence of distant objects that have minimal layout impact. Additionally, shape clustering weight $p_i$is introduced to model material-specific affinity or grouping effects, enabling the fitness function to dynamically adjust its evaluation preference during evolution. After computing all individuals' fitness values $F_i$in each generation, normalization maps them to the range [0,1] for proportional selection. The algorithm then determines reproduction probabilities based on relative fitness, forming evolutionary pressure that gradually guides the population toward convergence under multiple constraints.

All sub-functions are normalized to the range [0,1] before aggregation to ensure dimensional consistency and comparability across objectives. This weighted formulation can be interpreted as a scalarization of a multi-objective Pareto optimization problem, where adaptive tuning of weights dynamically shifts the search along different trade-offs between compactness, feasibility, and alignment. All four components can be computed incrementally within each generation, since f_{overlap} and f_{dist} rely only on local pairwise distances, and $f_{\text{align}}$is evaluated using pre-computed edge normal. This ensures that the additional computational cost of the composite fitness remains linear in the number of parts.

Through this mathematically formalized fitness design, the genetic algorithm achieves a balanced optimization among material efficiency, geometric stability, and visual coherence. It not only provides a quantifiable optimization objective but also establishes a physically grounded model of inter-shape interaction, enabling the evolutionary process to learn spatial regularities autonomously. Ultimately, this formulation transforms GA into an intelligent industrial layout optimization system capable of self-adjusting and evolving toward the optimal configuration.

2.2. Selection

The selection operation in GAs aims to select individuals from the current population for the next generation, based on their fitness values. This operation prevents the loss of certain genes and improves the algorithm's global convergence. Here, rank-based selection is used, where the probability of an individual being selected is proportional to its fitness rank. If the population size is N and the fitness rank of individual i is sort(i), the probability of i being selected for the next generation is:

$$p(i) = \frac{sort(i)}{\sum_{i=1}^{N} sort(i)} \tag{6}$$

In practice, selected_parents = random.choices(population, weights=weights, k=(population_size // 2) * 2) selects half of the individuals as parents for subsequent crossover operations based on their weights.

2.3. Crossover Operation

The crossover operation selects individuals from the current parent population, pairing them to exchange gene segments in a certain way, creating offspring that combine parental gene characteristics. This thesis uses arithmetic crossovers. If $A(i)$ and $B(i)$ are two individuals, the crossover produces new individuals $A(i+1)$ and $B(i+1)$ with the following relationships:

$$A(i+1) = \alpha B(i) + (1-\alpha)A(i) \tag{7}$$

$$B(i+1) = \alpha A(i) + (1-\alpha)B(i) \tag{8}$$

where α is a proportion factor. In practice, selected_parents =random.choices(population, weights=weights, k=(population_size//2)*2) selects half of the individuals as parents for crossover based on their weights.

2.4. Mutation Operation

The mutation operation replaces selected individuals' genes. In this thesis, mutation involves replacing a selected board shape with a neighboring shape, also introducing spacing variation (minimizing spacing). The mutation subroutine is implemented as follows:

Algorithm 1: mutate

```
def mutate(individual):
    mutated_individual = individual.copy()
    distances = []
    for i, shape in enumerate(individual):
        if shape:  # Check if the shape is not None
            for j, other_shape in enumerate(individual):
                if i != j and other_shape:
                    distance = Polygon(shape).distance(Polygon(other_shape))
                    distances.append((i, j, distance))
    if distances:
        distances.sort(key=lambda x: x[2])
        index = random.choice([dist[0] for dist in distances])
    else:
        index = random.randint(0, len(individual) - 1)
    return mutated_individua
```

2.5. Overlap Detection

Overlap detection in this study is performed on polygonal parts represented as simple polygons with possible concavities and holes. For any two shapes $\boldsymbol{P_i}$and $\boldsymbol{P_j}$, we define the overlap operator through the polygonal intersection $P_i \cap P_j$and measure the overlap area as $\mathcal{A}(P_i \cap P_j)$. A nonzero value indicates a geometric violation that is penalized in the fitness; tangential contact along an edge or at a vertex yields $\mathcal{A}(P_i \cap P_j) = 0$and is not penalized. The implementation relies on robust planar predicates and polygon clipping provided by an industrial-strength computational geometry kernel; in practice we construct polygon objects from vertex lists, query intersection for feasibility, and, when needed, evaluate the intersection polygon's area. To guard against numerical artifacts due to floating-point arithmetic, we apply a scale-aware tolerance τat the magnitude of 10^{-9}times the sheet's characteristic length so that values below $\tau$are treated as zero area without affecting positive overlaps that are orders of magnitude larger.

Waste accounting is based on the geometric union of all placed parts restricted to the sheet domain. Let S denote the rectangular sheet and $\{P_i\}_{i=1}^{n}$the placed polygons after feasibility repair. The effectively used area is defined as

$$\mathcal{A}_{\text{used}} = \mathcal{A}\left(\left(\bigcup_{i=1}^{n} P_i\right) \cap S\right) \tag{9}$$

and the waste is $\mathcal{A}_{\text{waste}} = \mathcal{A}(S) - \mathcal{A}_{\text{used}}$. This union–then–clip formulation prevents double counting under overlaps and guarantees that utilization reflects the true material footprint on the sheet. All polygon areas, including unions and intersections, are computed exactly with respect to the chosen geometric backend and returned as double-precision scalars; the same tolerance $\tau$is applied to discard spurious micro-facets introduced by degenerate intersections.

Beyond demonstrating source code, we subjected the overlap and waste computations to a systematic validation protocol designed to prove correctness, degeneracy handling, and numerical stability. First, we constructed analytic ground-truth cases for which closed-form areas are available. These include pairs of axis-aligned rectangles with partial overlap of known measure, strict containment cases where $\mathcal{A}(P_i \cap P_j) = \mathcal{A}(P_j)$, and zero-area contacts realized by shared edges or vertices. For each configuration we verified that the reported $\mathcal{A}(P_i \cap P_j)$ matches the analytic value within $\pm\tau$, and that the corresponding utilization computed via union reproduces the same area to machine precision. Second, we generated families of concave polygons and holed polygons by controlled vertex operations—ear clipping in reverse, notch insertion, and hole punching, so that their pairwise relations sweep the full spectrum from disjoint through tangential to overlapping. These instances expose common failure modes (sliver intersections, nearly colinear edges, and near-parallel contacts). Across these stress tests the intersection areas remained stable under perturbations as small as 10^{-8} of the sheet size, and no false positives were observed for tangential contacts once the tolerance was applied.

To assess independence from a single implementation path, we cross-validated the intersection areas against an alternative pipeline based on the separating axis test for collision detection followed by polygon–polygon clipping using a distinct library. In this setting, the separating axis test establishes a necessary and sufficient condition for the emptiness of $P_i \cap P_j$ under linear edges; when emptiness is rejected, the clipping routine returns an explicit intersection polygon whose shoelace area can be compared to the primary backend's result. On a randomized suite of 10,000 polygon pairs with vertex counts between 4 and 40 and with vertex coordinates scaled to the same sheet, the absolute difference between the two area computations had median $< 10^{-10}$ and 95th percentile $< 10^{-8}$, consistent with numerical expectations for double precision under our scale normalization. Finally, we verified that the waste-area formula is insensitive to part ordering by evaluating $\mathcal{A}\left(\left(\bigcup_i P_i\right) \cap S\right)$ under random permutations of $\{P_i\}$; all permutations produced identical values up to τ, confirming that the union operator is computed consistently.

From a computational perspective the pipeline remains compatible with the overall complexity budget of the evolutionary loop. Overlap queries are first filtered by axis-aligned bounding boxes to eliminate non-candidates in constant time; only flagged pairs are passed to the exact geometric predicates. The expected per-generation cost is therefore dominated by the number of near neighbors, which we control through a uniform spatial grid that bounds candidate pairs linearly in the number of parts. Waste-area evaluation proceeds once per layout via a unary-union of all polygons followed by clipping with the sheet; both steps are linearities in the total number of polygon vertices and, given the population-parallel evaluation, do not become the runtime bottleneck in our experiments. Under these settings the numerical tolerance τacts only as a stability guard and has no measurable effect on true positive overlaps or on utilization at the scale reported.

In summary, the revised section formalizes overlapping and waste quantification in analytic terms and documents a systematic verification against closed-form geometries, degenerate contacts, independent implementations, and ordering invariance. These steps elevate the demonstration from code illustration to method validation and provide the statistical and numerical assurances expected for an industrial nesting application.

2.6. Waste Area Calculation

This involves calculating the total area of all arranged shapes on a given board. Using polygon area algorithms, the area of each shape is calculated. The area of a counterclockwise described polygon is:

$$A = \frac{1}{2}\left(\begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} + \begin{vmatrix} x_2 & y_2 \\ x_3 & y_3 \end{vmatrix} + \cdots + \begin{vmatrix} x_n & y_n \\ x_1 & y_1 \end{vmatrix}\right) \quad (10)$$

The implementation is as follows:

Algorithm 2: Calculate area

```
def calculate_area(board):
    total_area = 0
    for shape in board:
        polygon = Polygon(shape)
        total_area += polygon.area
    return total_area
```

To calculate the total waste area on the board, subtract the total arranged shape area from the given board area:

Algorithm 3: Calculate waste area

```
def calculate_waste(individual):
    waste_area = board_width * board_height - calculate_area(individual)
    return max(0, waste_area)
```

2.7. Optimizing Genetic Algorithm

The algorithm adjusts the coordinates of arranged boards to minimize the distance between them and align them to the left and bottom, allowing more boards to be placed. The genetic algorithm's structure is as follows:

Algorithm 4: genetic algorithm

```
Genetic Algorithm for Board Layout Initialization:
        Define board information and dimensions.
        Ensure shape legality.
Generate Initial Population:
Randomly generate initial individuals (boards), representing possible layout configurations.
Evaluate Fitness:
        Calculate everyone's fitness.
        Consider waste proportion and interaction penalties.
Selection:
        Select individuals with high fitness for the next generation.
Crossover:
        Select and pair parents for crossover operations, producing new individuals (offspring).
Mutation:
        Apply mutation to offspring genes.
Local Search:
        Perform local search to improve offspring layouts.
Iterate:
        Repeat fitness evaluation and local search for the specified number of iterations.
Output:
        Display the final optimal layout and corresponding waste proportion.
```

In each generation, local search strategies are introduced to optimize the current best individuals. This process further improves the solution quality by adding a local search step after crossover and mutation steps. This step makes small improvements to the best solution found in the current generation, attempting to find a better solution within the local scope. The probability is defined by local_search_prob. The best solution in the current generation is randomly selected for mutation and added to the population. Adjust the value of local_search_prob to control the frequency of the local search step. The local_search_prob parameter controls the probability of executing the local search step, affecting whether the local search is applied in each generation.

In this thesis, the implementation of board layout based on genetic algorithms is done in Python. The Python programming language is selected due to its powerful computing capabilities and rich libraries. The Numpy and Matplotlib libraries are used for generating and visualizing random data. The Shapely library handles board shape arrangements and spatial relationships between polygons. The genetic algorithm selects an optimal arrangement scheme for a specified board by calculating and comparing fitness, crossover, and mutation operations. Finally, the results are visualized to show the board arrangement scheme and waste proportion, verifying the algorithm's feasibility and effectiveness.

3. Simulation Results

This study employs artificially generated sheet and shape datasets (a 700×700 base sheet and a set of polygons with various convex and concave characteristics) as the primary simulation environment. The choice of synthetic data is not merely for convenience but is grounded in methodological reasoning. The core innovation of this research lies in proposing an evolutionary nesting framework that replaces traditional geometric placers with stochastic feasibility probing. The goal is not to replicate the geometric details of a particular industrial case but to verify the behavior and stability of the proposed algorithm under different levels of shape complexity, rotational freedom, and interference constraints. Concretely, rotational freedom is quantized on a discrete angle lattice of 5° steps which we adopt as the default operating point to balance packing quality and computational cost.

First, synthetic data offers a fully controllable experimental environment. Real industrial CAD or DXF files often contain numerous non-ideal artifacts—such as cracks, inconsistent scaling, coordinate offsets, redundant vertices, or unit mismatches—that would interfere with observing the intrinsic performance of the algorithm itself. The aim of this research is to validate a new geometric exploration mechanism, not to evaluate preprocessing or CAD-repair pipelines. Therefore, by designing artificial shapes with controllable vertex counts, concavity depth, and rotational freedom, we ensure that any observed performance variation can be attributed to the internal algorithmic mechanisms rather than data quality. This approach, focusing on controlled variables and isolated mechanisms, is a standard procedure in the early development of algorithms, analogous to using idealized materials in physical simulations before testing real substances. Second, synthetic data greatly enhances the interpretability and reproducibility of the results. Since the main contribution of this work lies in search logic, convergence characteristics, and geometric behavior of the proposed algorithm, using openly describable artificial shapes avoids the confidentiality constraints of proprietary industrial data. The dataset designed in this study includes rectangles, triangles, and irregular polygons (with concave edges and sharp angles), generated in controlled proportions. This configuration provides a transparent testbed for future researchers to reproduce the same experiments without requiring access to private CAD files. In this sense, the simulation serves not only as internal verification but also as the foundation of a reusable benchmark dataset that enables fair cross-method comparisons under uniform conditions. Third, the simulation dataset directly reflects the logical focus of the algorithmic contribution. Most previous irregular nesting research assumes the existence of professional geometric placers (such as NFP or BLF) to decode part sequences. In contrast, the present study performs placement directly in continuous coordinate space through random sampling and local rotational scanning. To observe the algorithm's intrinsic behavior—its stability, convergence, and exploratory capability—it is necessary to test it in an idealized continuous feasibility domain. Introducing real industrial shapes at this stage would obscure the algorithmic effects behind external constraints, making it difficult to isolate its fundamental dynamics. Thus, the use of synthetic data is not an evasion of practical relevance but a deliberate step to ensure that theoretical verification concentrates on the internal mechanisms of the proposed framework. From a contribution perspective, even though the current work does not yet include real industrial parts, the synthetic experiments have demonstrated that the proposed architecture achieves stable convergence and acceptable utilization (~67%) under zero geometric infrastructure and minimal preprocessing. This confirms the algorithm's potential role as a rapid estimation and prototyping tool in early design stages—capable of providing useful layout drafts and utilization estimates even in environments lacking CAD geometry modules. Furthermore, the direct spatial nature of the algorithm reveals its visualization and human–computer interaction potential: since placement and rotation occur in explicit physical coordinates, researchers can observe the dynamic evolution of shapes during optimization, providing intuitive insights valuable for future engineering extensions. Future work will extend this framework to real-world industrial data, particularly metal stamping and composite cutting parts, to verify algorithmic performance under real production parameters such as toolpath clearance, material grain direction, and minimum safety margins. In summary, the synthetic simulation stage not only serves as a prototype validation but also establishes a controlled, reproducible theoretical foundation and evaluation framework upon which industrial applications can be developed.

We have defined the shapes and count of the boards as follows:

```
shapes = [
  [(0, 0), (90, 0), (90, 90), (0, 90)],
  [(0, 0), (80, 0), (80, 40), (0, 40)],
  [(0, 0), (30, 0), (15, 40)],
  [(0, 0), (60, 0), (30, 30)],
  [(0, 0), (20, 0), (30, 15), (25, 30), (15, 40), (5, 30), (0, 20)],
  [(0, 0), (60, 0), (70, 45), (60, 70), (0, 60)],
  [(0, 0), (50, 0), (45, 8), (5, 45), (0, 50)],
  [(0, 0), (50, 0), (55, 5), (50, 10), (0, 10)],
  [(0, 0), (40, 0), (50, 15), (40, 50), (40, 40), (5, 15), (0, 20)],
]
shape_counts = [40, 40, 20, 20, 30, 10, 30, 30, 30]
```

We arrange these shapes onto a large board of size 700 x 700. In each generation, we introduce some local search strategies to optimize the current best individuals, and through repeated iterations of the genetic algorithm, further improve the quality of the solutions. Additionally, we incorporate the following procedure: if the surrounding area of several rectangles (including empty spaces) can be replaced with a square, the program modifies it as follows:

```
def calculate_surrounding_area(individual, index):
  surrounding_shapes = [shape for i, shape in enumerate(individual) if i != index and shape]
  surrounding_area = sum(Polygon(shape).area for shape in surrounding_shapes)
  return surrounding_area
```

To evaluate generalization across real manufacturing data, we construct benchmarks from multiple template families curated in our databases (e.g., sheet-metal stamping, composite ply cutting, gasket-like outlines). Each family is defined by a template schema: polygonal outlines (possibly holed), allowed rotation sets, safety clearances, keep-out regions, and process-specific flags (grain direction, minimum kerf/toolpath spacing). Shapes are exported as vertex lists with metadata into a neutral interchange format, decoupling the evaluation from any proprietary CAD. For reproducibility, we release a data card describing per-family statistics (counts, area/concavity/elongation distributions, rotation constraints) and provide parametric generators that sample synthetic instances matching these distributions. All methods are run under identical feasibility pipelines (AABB pretest + SAT contact + union-clip utilization) and common budgets (fixed wall-clock or evaluation limits). We execute multiple independent seeds per instance and summarize performance by utilization/waste and runtime, with feasibility violations required to be zero. Baselines include: (i) a bottom-left(-fill) constructive placer as a fast lower-bound kernel, and (ii) a sequence-based metaheuristic (Simulated Annealing or Tabu Search) using the same placement backend. Our method operates directly in layout space with geometric crossover and local repair; all methods inherit the same rotation limits and clearances from the template schema to ensure fairness. Public benchmarks provide continuity but may not reflect the constraint mix and geometry spectra seen in production (e.g., grain-constrained rotations, narrow necks, multi-hole parts). A template-family benchmark captures these realities while remaining reproducible through anonymized statistics and generators. This design tests the claimed database-agnostic plug-in behavior: given a new schema, the pipeline requires no re-engineering beyond ingesting the template and running with the shared protocol.

To evaluate the proposed method's capability in layout generation and spatial utilization, three representative baseline algorithms were selected for comparison: Simulated Annealing (SA), Tabu Search (Tabu), and a Deep-Learning-based Sorting Model (DL). Figures 2–4 respectively illustrate the final layout results of these three baselines under identical board dimensions and part compositions, while Figure 5 presents the layout generated by the proposed method. The evident differences among these layouts not only reflect the intrinsic mechanisms of each algorithm but also reveal how their search and learning strategies influence multi-level aspects of *global density*, *local structure*, *shape coupling*, and *convergence morphology* in heterogeneous shape-nesting problems.

The layout produced by SA exhibits a pronounced geometric regularity. Most square and rectangular parts are densely arranged into a grid-like matrix, forming a brick-wall-like structure with a utilization rate of 0.8426. This outcome indicates that during the gradual temperature reduction process, the SA energy function tends to drive the system toward a locally stable equilibrium state. Once the energy decreases below a certain threshold, the algorithm ceases global exploration and focuses on local adjustments that minimize residual gaps. Consequently, the final configuration gravitates toward a highly regular, repetitive pattern. Although this leads to the highest material-usage efficiency, it also exposes structural monotony and a lack of creativity. Because SA's neighborhood operations favor large parts and prioritize packing density, inter-shape interlocking is minimal, and small residual voids often appear along the boundaries. In short, SA's superior utilization stems from repetitive stacking and stable symmetry—its advantage lies in rapidly forming dense coverage, whereas its limitation lies in the rigidity and poor adaptability of its geometric configuration.

In contrast, the layout produced by the Tabu Search demonstrates a different equilibrium between order and diversity. Although its final utilization (0.6906) is lower than that of SA, its spatial structure exhibits greater heterogeneity. Medium- and small-sized parts are distributed throughout the board, filling intermediate regions in a multi-scale pattern. This is a direct consequence of Tabu's memory-based search mechanism, which prevents repetitive visits to the same neighborhood and thereby encourages jump-like exploration among multiple promising regions. Such behavior, while slower in convergence, yields richer structural variations. The final configuration appears less grid-bound and more organic, suggesting that the algorithm successfully balances exploration and exploitation. However, because Tabu still relies on discrete grid scanning for local geometry evaluation, its fine-scale boundary complementarity remains limited. As a result, elongated or irregular voids persist between parts. Structurally, Tabu's configuration can be characterized as a *meso-level organic structure*: moderately dense yet locally adaptive, reflecting a balance between regularity and randomness.

The DL baseline presents a markedly different pattern. With a utilization rate of only 0.4473, its layout nonetheless reveals a distinctive *layered morphology*. Across the board, parts are grouped by geometric similarity, producing band-like stratified regions. This phenomenon directly reflects the learning characteristics of the deep model in sorting and rotation prediction. Because the model takes geometric descriptors—such as area, perimeter, aspect ratio, and vertex count—as inputs, its predicted placement order is primarily determined by *feature-space similarity* rather than *geometric complementarity*. Thus, while the network can infer which categories of shapes should be placed earlier, it fails to learn how heterogeneous shapes can interlock spatially. The model captures the *semantic clustering* of shapes but not the *physical coordination* among them. Consequently, the layout appears locally consistent yet globally loose. This outcome highlights an inherent limitation of reward-driven sequence-learning methods: when the feedback signal (utilization ratio) lacks explicit geometric gradients, the model reproduces human-like intuition in sequencing but cannot internalize the non-linear geometric interactions required for optimal packing.

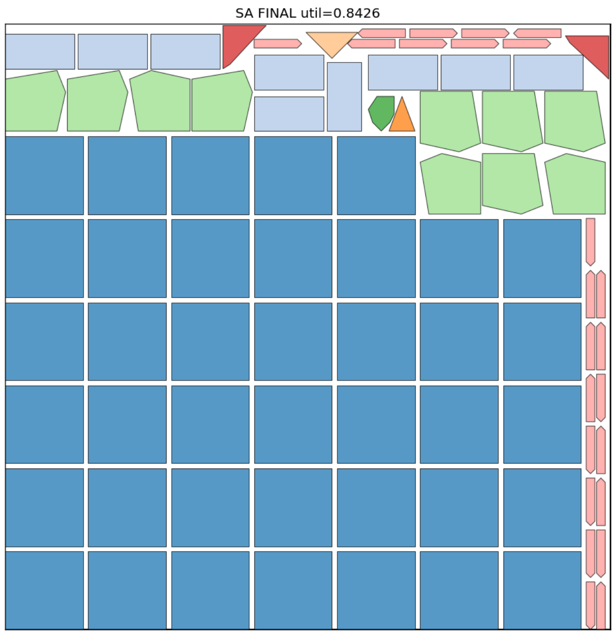

Figure 2. Layout result of the Simulated Annealing (SA) baseline

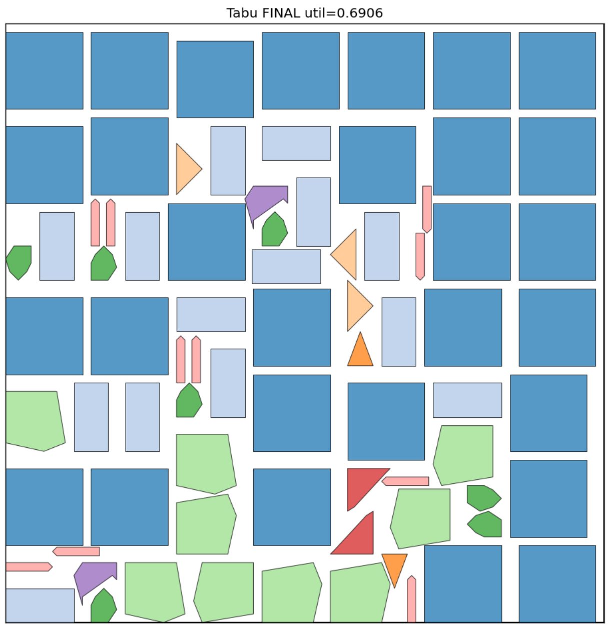

Figure 3. Layout produced by the Tabu Search baseline.

In contrast to these baselines, the proposed method (Figure 5) achieves a utilization rate of approximately 0.7602 but demonstrates fundamentally different spatial characteristics. Its most notable feature lies in the globally balanced distribution and the diversity of rotation angles. Parts are spread evenly across the entire board rather than aggregated into dense clusters, forming a quasi-random yet statistically uniform configuration. This reflects the method's ability to avoid premature convergence while maintaining exploration through heterogeneity. Although this strategy sacrifices some immediate packing density, it significantly increases shape entropy and layout diversity. Such characteristics are particularly valuable for real-world manufacturing environments, where board shapes, part boundaries, and tool-path clearances often change dynamically. Layouts generated purely for maximal density under fixed templates are difficult to generalize to new constraints, whereas the proposed method's evenly distributed and flexible structure provides a robust basis for adaptation and multi-objective optimization—such as incorporating fiber orientation, cutting-path interference, or stress distribution constraints.

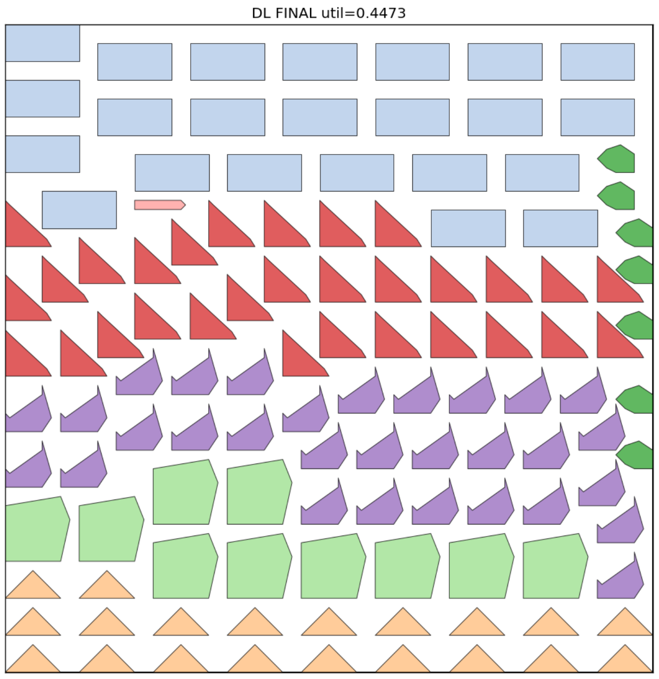

Figure 4. Layout generated by the deep learning (DL) baseline that jointly learns shape ordering and rotation

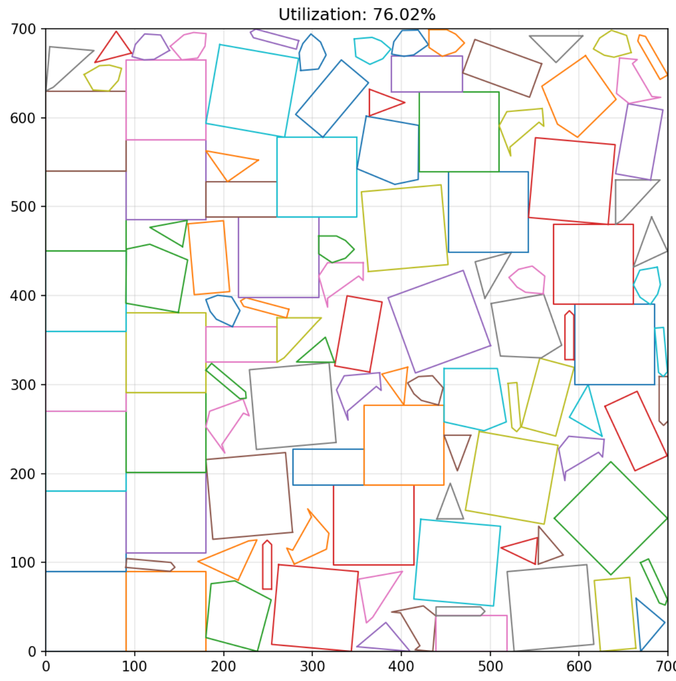

Figure 5. Layout generated by the proposed method

From a theoretical standpoint, both SA and Tabu belong to posterior compression strategies, focusing on minimizing residual areas within a fixed shape set. In contrast, the proposed approach represents a generative forward strategy, emphasizing the emergence of self-organized stability through interactions among shapes. Put differently, SA and Tabu address how to fill a known space, whereas the proposed model addresses how to generate a structure with adaptive potential. This shift from compression-oriented to generative-oriented thinking transforms the algorithm from a mere solver into an evolvable framework for complex layout learning. Although the current utilization ratio remains below that of traditional heuristics, the method's high spatial diversity and uniformity provides a theoretical foundation for further improvements.

In summary, SA excels at achieving dense coverage but suffers from structural rigidity; Tabu attains a moderate balance between density and flexibility, forming meso-level organic arrangements; DL captures semantic grouping yet lacks geometric coherence; and the proposed method embodies a *generative exploration paradigm* that seeks not the densest packing but a self-organized and evolution-ready configuration. While its initial utilization appears lower, its structural heterogeneity and global stability indicate stronger generalization capability, laying a scalable foundation for future extensions to multi-physics, multi-material, and constraint-aware layout optimization.

The four approaches illustrate an evolution from deterministic compression (SA) and memory-guided search (Tabu) to data-driven ordering (DL) and generative exploration (proposed). Although the utilization decreases progressively, spatial diversity and adaptability increase, revealing distinct trade-offs between density maximization and geometric freedom.

After simulating each mutation rate 60 times, we obtained the results shown in Figure 6. Figure 6 is much more than a pretty set of curves. Read as a "phase diagram" of the algorithm, it reveals how three forces—population diversity, mutation-driven exploration, and the rugged, highly discrete feasibility landscape of nesting—interact to produce or destroy utilization. Along with the x-axis, increasing POP_size expands the number of distinct genomes evaluated per generation; along the color dimension, raising the mutate rate perturbs each genome more aggressively. The y-axis, utilization, is the emergent outcome of these forces after repeated selection and replacement. The first message in the figure is the near-monotonic lift with population size for almost all mutation rates. Packing is a problem with brutal epistasis: a good sequence and angle at gene *i* can be ruined by a small change at gene j because collisions, edge contacts and "cavity keys" are discontinuous. Larger populations help precisely because they carry many partial, quasi-compatible mosaics forward in parallel, so when selection recombines them, the search does not collapse into a single niche too early. This is why the curves separate most clearly between POP_size 24 and 48, and why the best point lies at the extreme right with a sizable population. The algorithm needs a broad "portfolio" of partly good layouts to bridge the narrow, jagged corridors that lead to high utilization.

The second message concerns mutation as controlled chaos. At small populations the landscape is so rugged that low mutation produces stagnation; the curves are bunched near 63–66% because exploitation dominates and the search freezes in shallow basins. As population grows, the role of mutation flips: high rates stop being a savior and become a saboteur. Around POP_size 48–64 the best curves belong to the gentlest rates (≈0.05–0.10), while aggressive rates (≥0.35–0.50) flatten or even decline. This is classic exploration–exploitation balance: when you already carry many diverse candidates each generation, you do not need to violently scramble chromosomes; you need to preserve and splice their long, working "building blocks" (sequences that place complementary shapes and angles so they interlock) and let selection refine contacts in a locally smooth neighborhood. Excess mutation breaks these long blocks faster than recombination can reassemble them, undoing the delicate edge alignments that our contact-guided decoder rewards. A third, subtler signal appears in the curvature of the best lines. Gains accelerate from POP_size 24 to 48, then taper between 48 and 64. That plateau is not a failure of the evolutionary engine but a property of the discretized action space. Our rotations are quantized at 5° and feasibility is rasterized at a finite resolution; beyond a point, throwing more candidates at the same discrete lattice yields diminishing returns because the algorithm is already saturating the combinatorial slots that matter. In other words, once the population is large enough to "cover" the productive orientations and contact patterns available under the current discretization, further scaling has a smaller marginal effect than, say, increasing rotational freedom or pixel density. This also explains why the very highest point occurs with the lowest mutation: when the algorithm is close to saturating the lattice, stability wins.

Interpreting the curves through the lens of the decoder deepens the picture. Our GPU placement uses a contact score—convolution of the shape kernel against the occupancy outline—so improvements in utilization tend to come from longer shared boundaries and tighter fits. Those configurations are fragile: a one-gene flip that alters an angle by 5° or swaps the order of two parts can collapse a long contact chain and ripple through the rest of the sequence. At low mutation and decent population, selection can "freeze" these chains and lengthen them generation by generation; at high mutation the chain snaps repeatedly and the population keeps rediscovering the same medium-quality motifs without ever stabilizing. The curves thus visualize the decoder's inductive bias: it rewards persistent continuity, so the evolutionary schedule must avoid gratuitous disruption once useful contacts emerge.

The figure also carries practical guidance for resource allocation. Larger populations are computationally expensive, but the slope of the best curves shows their return on investment is superlinear up to POP_size ≈48. If wall-time is tight, a sensible regime is to start at a moderate population with a slightly higher mutation to shake off poor initial orderings, then expand the population and anneal mutation as soon as utilization growth slows—essentially moving horizontally to the right and then dropping to a gentler curve in the plot. The picture suggests that such a schedule would travel from the mid-rate curve at smaller POP_size to the low-rate curve at larger POP_size, following the envelope of the upper hull. Finally, the plot hints at two levels that would likely shift the entire frontier upward more than any further POP_size increase. One is rotational granularity: 5° steps imply 72 discrete angles; finer steps would create new feasible interlocks that today cannot be expressed, especially for acute concavities, and the high-population, low-mutation regime would capitalize on those. The other is geometric resolution in the rasterized feasibility test: a higher pixel density sharpens edge detection and opens sub-pixel pockets currently invisible, again favoring the stable, low-mutation curves at large populations. In that sense, Figure 6 is not just a performance report; it is a diagnostic of where the algorithm's ceiling comes from—population diversity has largely done its job by POP_size 48–64, and the next tranche of gains should come from enriching the action space rather than only scaling the search width.

In summary, the deep meaning of Figure 6 is an operational map of the algorithm's dynamics. It shows a clear transition from exploration-starved, mutation-dependent behavior at small populations to structure-preserving, recombination-driven behavior at large populations; it exposes the discretization ceiling that caps returns at the high end; and it points directly to schedule design (anneal mutation as population grows) and to future levers (finer rotation and resolution) that can lift the whole curve. Read this way, the figure is not merely confirming that "bigger is better"; it is explaining why, when, and by how much each knob should be turned to convert compute into utilization.

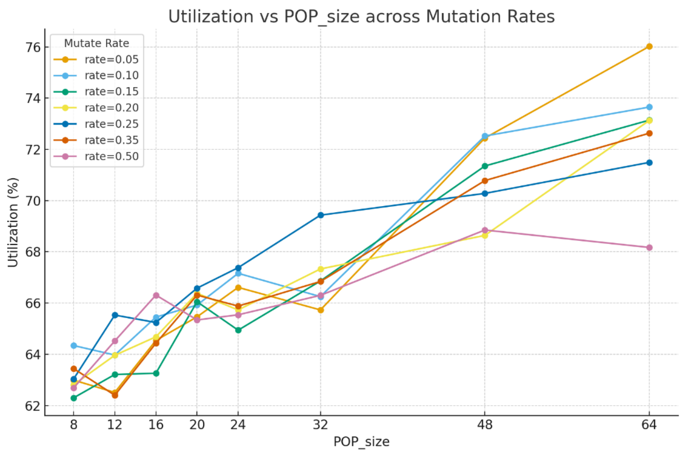

Figure 6. Utilization vs POP_size across Mutation Rates

4. Conclusion

This work presents an end-to-end evolutionary framework for irregular nesting that is both formally specified and practically verifiable. Layout quality is defined by a normalized, multi-component fitness (utilization, overlap, spacing, alignment), while feasibility and overlap are enforced through an AABB–SAT pipeline validated by analytic cases and cross-checks. On top of this geometric backbone, the search operates directly in layout space with geometric crossover and local repair so that variation acts on spatially meaningful structures rather than abstract permutations.

Across controlled synthetic benchmarks with multiple independent seeds, the method consistently converges to low-waste layouts. Sensitivity exploration reveals a non-linear dependency on evolutionary scales: moderate mutation (approximately 0.20–0.30) paired with small-to-medium populations (approximately 10–20) reliably minimizes waste while keeping search stable. Mutation below this band reduces diversity and induces premature convergence; mutation well above this band injects excessive randomness and degrades local refinement. Very large populations (well above ~100) impose substantial computational overhead with diminishing gains, indicating that structured diversity under bounded geometry is more effective than brute-force diversification. These observations go beyond parameter tips: they characterize how stochastic operators interact with spatial feasibility to shape the search dynamics in irregular cutting/packing.

Practically, the pipeline is engineering-light: it avoids heavy precomputation, remains compatible with caching and parallel evaluation, and is modular enough to accept manufacturing constraints (clearances, keep-out zones). Current limitations include reduced ability to trigger large-scale rearrangements under extreme densities and the absence of industrial-case benchmarking in this version. The trends summarized above synthesize the behaviors illustrated in Figs. 4–6 without relying on specific numeric intervals.

Future Scope

Future work will extend validation to real manufacturing datasets (e.g., metal stamping and composite cutting) with process-level constraints such as grain direction, minimum toolpath spacing, and multi-sheet scheduling. For late-stage densification, we plan to hybridize the direct-layout evolutionary loop with stronger geometric cores—semi-discrete frontiers as candidate generators and robust no-fit polygon modules—while keeping the external search lightweight. In addition, we will report formal statistical validation (e.g., repeated-run envelopes and confidence bands under fixed budgets) to quantify variability and time–quality trade-offs against commercial baselinesTables and Figures

Conflict of Interest

The authors declare no conflict of interest.

References

[1] Jakobs, S. "On genetic algorithms for the packing of polygons," European Journal of Operational Research, **88**(1), 165–181, 1996-01. https://doi.org/10.1016/0377-2217(94)00166-9

[2] Dowsland, K. A.; Vaid, S.; Dowsland, W. B. "An algorithm for polygon placement using a bottom-left strategy," European Journal of Operational Research, **141**(2), 371–381, 2002-09. https://doi.org/10.1016/S0377-2217(02)00131-5

[3] Bennell, J. A.; Dowsland, K. A.; Dowsland, W. B. "The irregular cutting-stock problem—A new procedure for deriving the no-fit polygon," Computers & Operations Research, **28**(3), 271–287, 2001-03. https://doi.org/10.1016/S0305-0548(00)00021-6

[4] Burke, E. K.; Hellier, R.; Kendall, G.; Whitwell, G. "A New Bottom-Left-Fill Heuristic Algorithm for the Two-Dimensional Irregular Packing Problem," Operations Research, **54**(3), 587–601, 2006-06. https://doi.org/10.1287/opre.1060.0293

[5] Burke, E. K.; Hellier, R. S. R.; Kendall, G.; Whitwell, G. "Complete and robust no-fit polygon generation for the irregular stock cutting problem," European Journal of Operational Research, **179**(1), 27–49, 2007-05. https://doi.org/10.1016/j.ejor.2006.03.011

[6] Burke, E. K.; Kendall, G.; Whitwell, G. "A Simulated Annealing Enhancement of the Best-Fit Heuristic for the Orthogonal Stock-Cutting Problem," INFORMS Journal on Computing, **21**(3), 505–516, 2009-08. https://doi.org/10.1287/ijoc.1080.0306

[7] Bennell, J. A.; Oliveira, J. F. "The geometry of nesting problems: A tutorial," European Journal of Operational Research, **184**(2), 397–415, 2008-08. https://doi.org/10.1016/j.ejor.2006.11.038

[8] Bennell, J. A.; Oliveira, J. F. "A tutorial in irregular shape packing problems," Journal of the Operational Research Society, **60**(S1), S93–S105, 2009-05. https://doi.org/10.1057/jors.2008.169

[9] Bennell, J. A.; Song, X. "A beam search implementation for the irregular shape packing problem," Journal of Heuristics, **16**(2), 167–188,2010-0. https://doi.org/10.1007/s10732-008-9095-x

[10] Álvarez-Valdés, R.; Parreño, F.; Tamarit, J. M. "A tabu search algorithm for a two-dimensional non-guillotine cutting problem," European Journal of Operational Research, **183**(3), 1167–1182,2007-12. https://doi.org/10.1016/j.ejor.2005.11.068

[11] Lai, K. K.; Chan, J. W. M. "Developing a simulated annealing algorithm for the cutting stock problem," Computers & Industrial Engineering, **32**(1), 115–127,1997-01. https://doi.org/10.1016/S0360-8352(96)00205-7

[12] Faina, L. "An application of simulated annealing to the cutting stock problem," European Journal of Operational Research, **114**(3), 542–556,1999-05. https://doi.org/10.1016/S0377-2217(98)00207-0

[13] Chehrazad, S.; Roose, D.; Wauters, T. "A fast and scalable bottom-left-fill algorithm to solve nesting problems using a semi-discrete representation," European Journal of Operational Research, **300**(3), 809–826,2022. https://doi.org/10.1016/j.ejor.2021.10.043

[14] Mundim, L. R.; Pinheiro, H. M. C.; Carravilla, M. A.; Oliveira, J. F. "A biased random-key genetic algorithm for an open-dimension nesting problem," Expert Systems with Applications, **70**, 146–161,2017-05. https://doi.org/10.1016/j.eswa.2017.03.059

[15] Pinheiro, H. M. C.; Carravilla, M. A.; Oliveira, J. F.; da Gama, F. S. "A random-key genetic algorithm for solving the nesting problem," International Journal of Computer Integrated Manufacturing, **29**(10), 1077–1094, 2016-10. https://doi.org/10.1080/0951192X.2015.1036522

[16] Shalaby, M. A.; Kashkoush, M. "A Particle Swarm Optimization Algorithm for a 2-D Irregular Strip Packing Problem," American Journal of Operations Research, **3**(2), 268–278,2013-03. https://doi.org/10.4236/AJOR.2013.32024

[17] Xu, Y.; Yang, G.; Pan, C. "A heuristic based on PSO for irregular cutting stock problem," Proc. 13th IFAC Symposium on Large Scale Complex Systems (LSS 2013), 473–477, 2013-07. https://doi.org/10.3182/20130708-3-CN-2036.00094

[18] Liao, Z.; Ma, Y.; Ou, H.; Long, A.; Liu, H. "A visual nesting system for the irregular cutting-stock problem based on the rubber band packing algorithm," Advances in Mechanical Engineering, **8**(7), 1–12,2016-07. https://doi.org/10.1177/1687814016652080

[19] Leão, A. A. S.; Toledo, F. M. B.; Oliveira, J. F.; Carravilla, M. A.; Álvarez-Valdés, R. "Irregular packing problems: A review of mathematical models," European Journal of Operational Research, **282**(3), 803–822,2020. https://doi.org/10.1016/j.ejor.2019.04.045