HPC Simulated Annealing for Non-Convex Polygon Packing in a Square

Bracketing, Bisection Refinement, Time-Limited Shrink Polish, and Sweep Diagnostics

Joel Amir D. Maldonado Tänori

Applied Mathematics PhD, University of Arizona

January 15, 2026

The Shape We Pack

Agenda

1. Problem and constraints
2. Geometry model: triangulation + SAT + broad-phase rejects
3. Energy function, feasibility, and SA moves
4. Outer optimizer: bracketing + bisection
5. Time-limited polish: adaptive shrink search + checkpoints
6. HPC reliability: determinism, file safety, GLIBC considerations
7. Sweep diagnostics: density, boundary effects, disorder statistics
8. Best-solution visuals

Problem Statement

Goal. Pack N identical copies of a fixed **non-convex** polygon $P \subset \mathbb{R}^2$ into a square of side length L.

Decision variables (per instance $i = 1, \ldots, N$):

$$\left(c_x^{(i)},\, c_y^{(i)},\, \theta^{(i)} \right) \in \mathbb{R}^2 \times [0, 2\pi)$$

where $(c_x^{(i)}, c_y^{(i)})$ is translation and $\theta^{(i)}$ is rotation.

Constraints.

- **Non-overlap:** interiors of instances are disjoint.
- **Containment:** each polygon lies inside the square $[-L/2, L/2]^2$.

Objective. Minimize L (tightest feasible packing).

A Provable Lower Bound (Area Bound)

Let $\mathrm{area}(P)$ be the polygon area. Then any feasible packing must satisfy:

$$L^2 \;\geq\; N \cdot \mathrm{area}(P) \quad \Rightarrow \quad L \;\geq\; \sqrt{N\,\mathrm{area}(P)}.$$

In code, we keep a strictly provably infeasible threshold:

$$L_{\text{area}} = \sqrt{N\,\mathrm{area}(P)}, \qquad L_{\text{area-infeas}} = (1-\varepsilon)\,L_{\text{area}}.$$

Interpretation. If $L \leq L_{\text{area-infeas}}$, the instance is **provably infeasible** (no need to run SA).

Geometry Model: Triangulation + SAT

We represent the non-convex polygon P by:

- A fixed vertex list $\{v_k\}_{k=1}^{N_V}$ in local coordinates.
- A fixed triangulation $\{\Delta_t\}_{t=1}^{N_T}$, each triangle uses indices into the vertex list.

Each instance i produces world vertices:

$$w_k^{(i)} = R(\theta^{(i)})\, v_k + \begin{bmatrix} c_x^{(i)} \\ c_y^{(i)} \end{bmatrix}.$$

Collision test between instances (i, j):

1. Broad-phase AABB overlap
2. Broad-phase bounding-circle reject
3. Narrow-phase triangle-triangle SAT for all (t_a, t_b)

Broad-Phase Acceleration

AABB reject. If axis-aligned bounding boxes do not overlap, polygons cannot overlap.

Bounding-circle reject. Precompute base radius r:

$$r = \max_k \|v_k\|_2.$$

If centers are far:

$$\|c^{(i)} - c^{(j)}\|_2 > 2r \quad \Rightarrow \quad \text{no overlap.}$$

Uniform grid (spatial hashing). Each instance belongs to one grid cell; collision checks only consider neighbor cells within a radius based on $2r$.

Outcome. Pairwise overlap checks scale with *local neighborhood* interactions rather than $O(N^2)$ in practice.

Containment Penalty via AABB

Containment is enforced through an **outside penalty** computed from AABB vs. square:

$$\mathrm{out}_i(L) = \sum_{\text{violations}} d^2$$

where d is how far the AABB exceeds $\pm L/2$ in any direction.

Benefit. Cheap to compute, robust, and effective when combined with increased penalty weights in Phase B.

Energy Function and Feasibility Metric

We separate **energy** (for SA acceptance) from **feasibility** (for outer logic).

Totals:

$$\mathrm{ov}(L) = \sum_{i<j} \mathrm{overlap\_penalty}(i,j), \qquad \mathrm{out}(L) = \sum_{i} \mathrm{outside\_penalty}(i).$$

Energy (SA at fixed L):

$$\mathcal{E} = \lambda\,\mathrm{ov} + \mu\,\mathrm{out}$$

(with weights scheduled by phase).

Feasibility metric:

$$\mathrm{feas} = \mathrm{ov} + \mathrm{out}.$$

We declare "feasible" if $\mathrm{feas} \le \tau$ for a small tolerance τ and L is not area-provably infeasible.

Move Set and Incremental Updates

Each SA iteration chooses a random index k and applies one of:

- **Reinsert (small probability):** randomize (c_x, c_y, θ) uniformly.
- **Local jitter:** $(c_x, c_y) \leftarrow (c_x, c_y) + \Delta$, $\Delta \sim \mathrm{Unif}([-s, s]^2)$.
- **Rotation mix:** with probability p_{rot}, $\theta \leftarrow \theta + \Delta\theta$.

Incremental bookkeeping. Only terms involving instance k are recomputed:

$$\mathrm{ov} \leftarrow \mathrm{ov} + (\mathrm{ov}_k^{\mathrm{new}} - \mathrm{ov}_k^{\mathrm{old}}), \qquad \mathrm{out} \leftarrow \mathrm{out} + (\mathrm{out}_k^{\mathrm{new}} - \mathrm{out}_k^{\mathrm{old}}).$$

Outcome. Fast inner loop; suitable for HPC sweeps over many N.

Two-Phase SA Schedule

We use two sequential phases at fixed L:

Phase A (Explore).

- Higher temperature $T_{\text{start}} \to T_{\text{end}}$
- Larger step sizes
- Moderate penalties (λ, μ)
- Purpose: escape poor initializations and reduce gross overlaps/outside

Phase B (Enforce).

- Lower temperatures
- Smaller step sizes
- **Penalty ramp:** $(\lambda, \mu) \leftarrow \min((\lambda, \mu) \cdot \rho,\ (\lambda_{\max}, \mu_{\max}))$
- Purpose: aggressively drive $\mathrm{feas} \to 0$

SA Acceptance Rule

Given current energy $\mathcal{E}$ and proposed energy $\mathcal{E}'$, accept with:

$$\Delta\mathcal{E} = \mathcal{E}' - \mathcal{E}, \qquad \mathbb{P}(\text{accept}) = \begin{cases} 1, & \Delta\mathcal{E} \le 0, \\ \exp(-\Delta\mathcal{E}/T), & \Delta\mathcal{E} > 0. \end{cases}$$

We track the best feasibility configuration during the run:

$$\text{feas}_{\text{min}} = \min_{t} \text{feas}(t),$$

and restore to that best configuration at the end of each trial.

Simulated Annealing at Fixed L (Clean Skeleton)

State

$$x = \{(c_x^{(i)}, c_y^{(i)}, \theta^{(i)})\}_{i=1}^{N}$$

Energy and Feasibility

$$\mathcal{E}(x; L) = \lambda \operatorname{ov}(x) + \mu \operatorname{out}(x; L), \qquad \operatorname{feas}(x; L) = \operatorname{ov}(x) + \operatorname{out}(x; L)$$

Algorithm Skeleton

1. Initialize x (grid init or warm-start)
2. Phase A (explore): larger moves, higher temperature
3. Phase B (enforce): smaller moves, ramp penalties
4. Track best snapshot (minimum feas) and return it

Outer Loop Overview

We minimize L via:

1. **Initialize** L from a grid layout (near-square arrangement).
2. **Bracketing:** shrink until infeasible (or grow until feasible) to find $[L_{\text{low}}, L_{\text{high}}]$.
3. **Bisection:** refine the bracket to a tight feasible L.
4. **Polish:** time-limited stochastic descent to shrink further.

Reliability constraints. The area bound prevents invalid brackets; best configurations are carried across L updates by scaling warm-starts.

Bisection Refinement

Given $L_{\text{low}} < L_{\text{high}}$:

$$L_{\text{mid}} = \tfrac{1}{2}(L_{\text{low}} + L_{\text{high}}).$$

If $L_{\text{mid}} \leq L_{\text{area-infeas}}$: mark infeasible without SA.

Else:

- Warm-start by scaling best-feasible configuration from L_{high} to L_{mid}.
- Run bounded SA trials at L_{mid}.
- If feasible: $L_{\text{high}} \leftarrow L_{\text{mid}}$; else $L_{\text{low}} \leftarrow L_{\text{mid}}$.

After k steps, bracket width shrinks by 2^{-k}.

Polish Stage: Adaptive Shrink Search

After bisection, we have a strong feasible configuration at $L^\star$.

Attempt a shrink:

$$L_{\text{try}} = L^\star(1 - \epsilon)$$

Scale positions, then run a few SA trials.

Adaptive control. Tune ϵ based on recent success rate (stochastic descent with backoff) until time limit is reached.

Time Limit, Checkpoints, and SIGTERM

Time-limited polish. We cap polish runtime (e.g., 900s) to guarantee completion in Slurm arrays.

Periodic checkpoints (every Δt seconds):

- `csv/<prefix>_checkpoint_Nxxx.csv`
- `img/<prefix>_checkpoint_Nxxx.svg`

SIGTERM handling. Near walltime Slurm sends SIGTERM; the handler:

- flushes **best snapshot** to best and checkpoint files,
- exits cleanly.

Outcome. Even canceled jobs produce usable artifacts.

HPC Engineering: Determinism and File Safety

Deterministic seeds. Each trial seed is derived from:

$$\text{seed} = f(\text{base\_seed},\ \text{run\_id},\ \text{trial\_id})$$

(e.g., SplitMix64 diffusion) for reproducibility across array jobs.

Unique output prefixes.

$$\texttt{out\_prefix} = \texttt{N\{N\}\_job\{job\}\_task\{task\}}$$

prevents file collisions under concurrency.

Build environment. Compile on compute nodes (or use cluster toolchain) to avoid GLIBC mismatch between login/desktop and compute nodes.

Complexity Notes and Practical Performance

Worst case. Collision evaluation can be heavy, but practice is improved by:

- grid neighbor enumeration (local pairs),
- AABB + bounding-circle rejects (broad phase),
- small fixed triangulation size (N_T constant),
- incremental updates per move (only one instance changes).

Scaling expectation. Runtime grows with local density and neighbor interactions, typically milder than $O(N^2)$.

Overnight HPC Sweep: Execution Summary

What Was Run

A Slurm job array solved **200 independent packing problems**, one per $N = 1, \ldots, 200$, using identical solver logic and fixed per-task budgets.

Resource Budget

- Up to **20 tasks in parallel** (`#SBATCH --array=1-200%20`)
- **30 minutes wall time per task** (`#SBATCH --time=00:30:00`)
- 1 CPU core, 2 GB memory per task

Time Allocation Within Each Task

- **Bracketing + bisection:** bounded SA trials (designed to always finish)
- **Polish stage:** time-limited to **12 minutes**
- **Checkpoints:** every 3 minutes during polish
- **Termination safety:** SIGTERM handler flushes best artifacts before exit

Parallelization Model

Type of Parallelism

Embarrassingly parallel parameter sweep. Each value of N is solved independently:

$$\text{Task } N \quad \longleftrightarrow \quad \text{one SLURM array element.}$$

Why This Scales Well

- No inter-task communication or synchronization
- No MPI overhead; pure throughput scaling
- Fault isolation: failure at one N does not affect others

Scheduling Benefits

- Slurm load-balances variable runtimes across N
- Concurrency cap prevents queue flooding
- Natural extension to longer per-N runs or replicated runs per N

Reproducibility Under Parallel Execution

Deterministic Seeding

Each task uses a fixed base seed combined with:

$$\text{seed} = f(\text{base seed},\ N,\ \text{trial id}),$$

ensuring reproducibility regardless of execution order or node placement.

File Safety

- Unique output prefixes: `N{N}_job{job}_task{task}`
- No collisions under concurrent tasks
- Safe output on shared filesystems

Reliability

- SIGTERM delivered before wall time (buffer) to flush artifacts
- Periodic checkpoints reduce risk of losing progress

Sweep Outputs and Aggregation

For each N, the code writes:

- `img/<prefix>_best_Nxxx.svg` and `csv/<prefix>_best_polys_Nxxx.csv`
- `img/<prefix>_checkpoint_Nxxx.svg` and `csv/<prefix>_checkpoint_Nxxx.csv`

A sweep analysis script aggregates all best CSVs into:

- `analysis/summary.csv` (one row per N)
- `analysis/plots/*.png` (diagnostic graphs)

These plots quantify **how tight** the packing is and **why** larger N can become loose.

Key Packing Quality Metrics

Let $\mathrm{area}(P)$ be polygon area and $L(N)$ be best feasible side length for size N.

Density (primary):

$$\rho(N) = \frac{N\,\mathrm{area}(P)}{L(N)^2} \qquad \text{(higher is better)}$$

Empty area (slack):

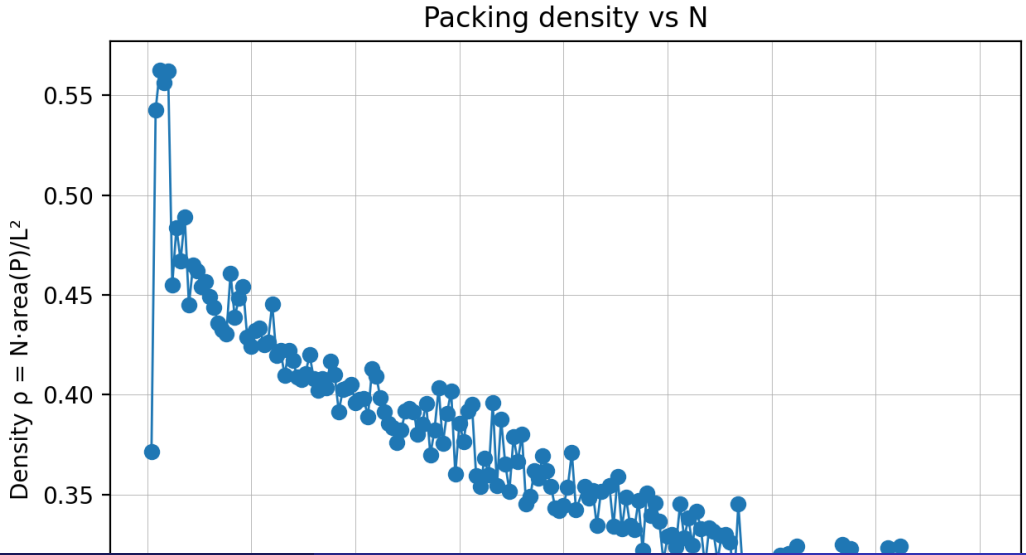
$$\Delta A(N) = L(N)^2 - N\,\mathrm{area}(P)$$

Boundary fraction (boundary waste proxy): fraction of centers within distance kr of the square boundary, where r is the polygon bounding radius.

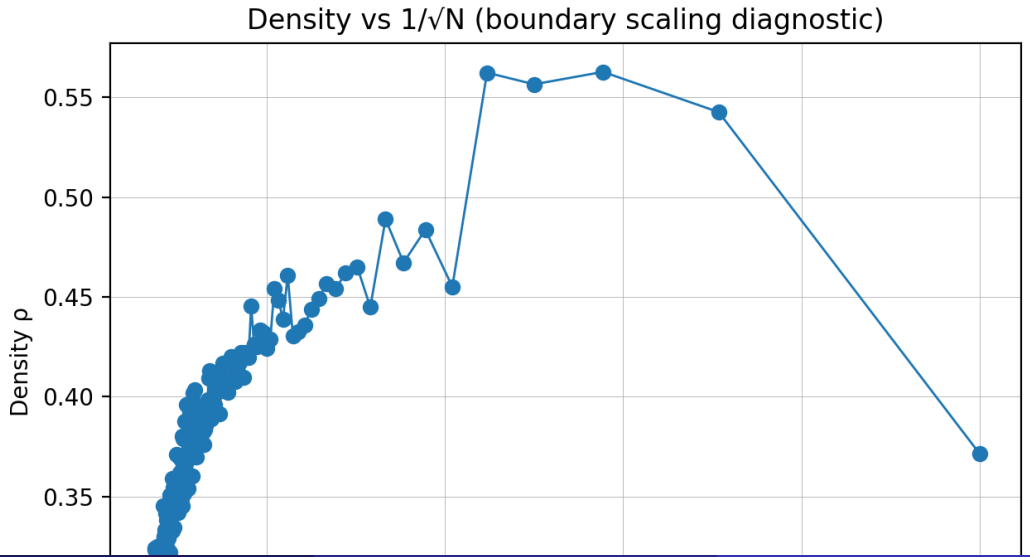
Structure/disorder:

- nearest-neighbor distance statistics (mean / std),
- orientation entropy of $\{\theta_i\}$ (high entropy $\Rightarrow$ amorphous).

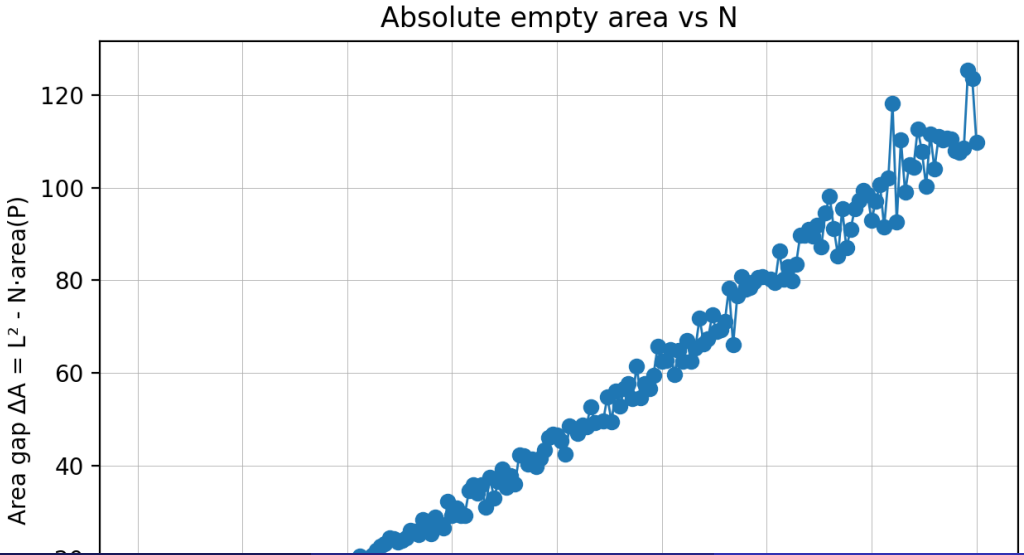
Density vs N



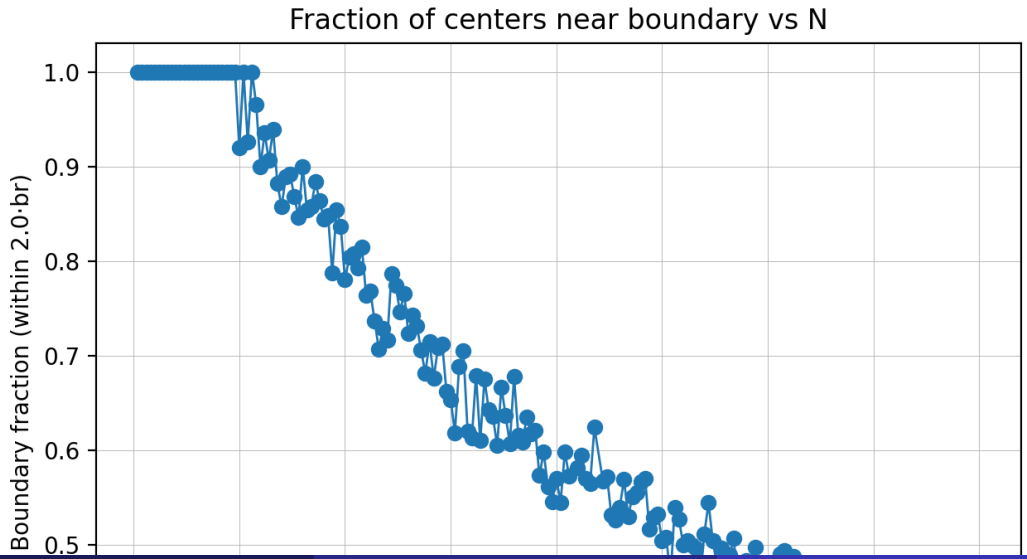
Density vs $1/\sqrt{N}$ (Boundary Scaling Diagnostic)



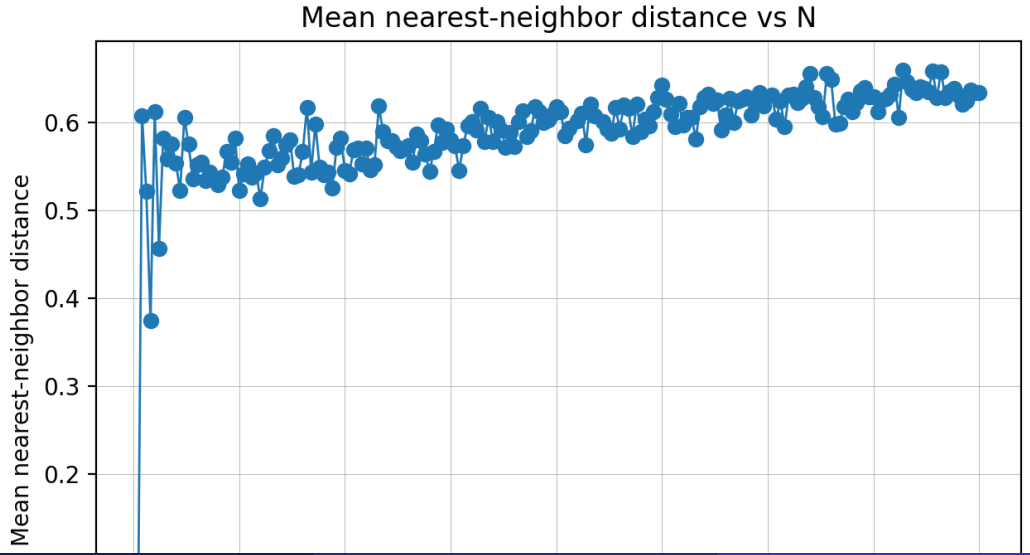
Empty Area (Slack) vs N



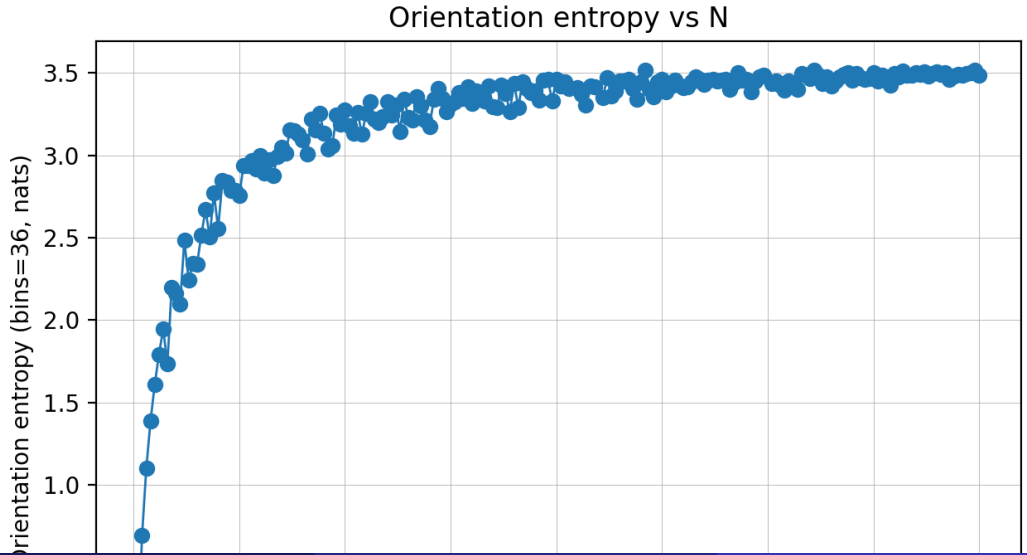
Boundary Fraction vs N



Nearest-Neighbor Spacing vs N



Orientation Entropy vs N



Interpreting the Diagnostics

Case A: boundary-limited (good interior, bad boundary).

- density vs $1/\sqrt{N}$ approximately linear,
- interior spacing stable, but boundary fraction high.

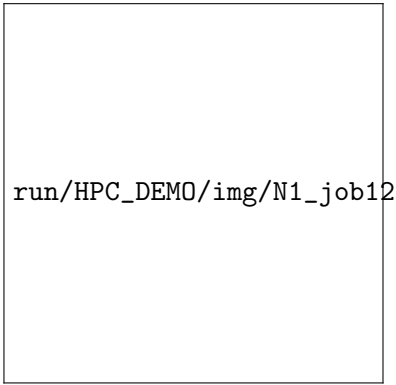
Response: warm-start from a periodic tiling; fit near-square patches; boundary-aware refinement.

Case B: motif failure (interior also loose).

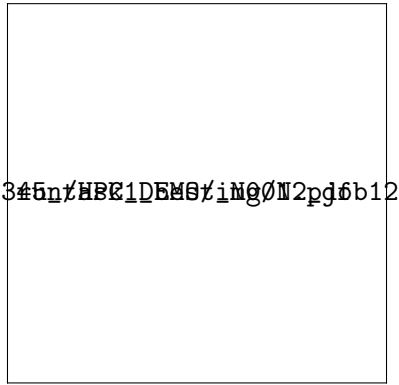
- density flattens too low,
- NN mean grows and orientation entropy remains high.

Response: discover small periodic motifs (periodic boundary SA), then tile; add structured moves / templates.

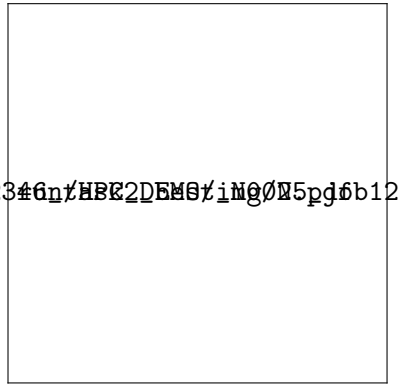
Best Solutions Across N (Small)



$N = 1$

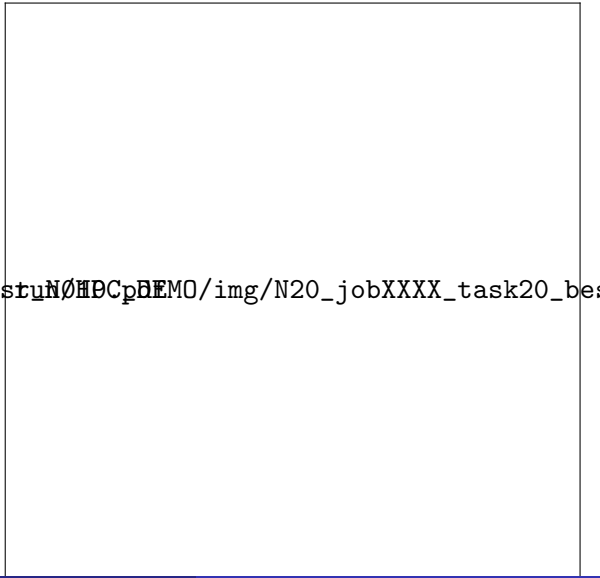


$N = 2$



$N = 5$

Best Solutions Across N (Medium)



Best Solutions Across N (Large)

Best Solution at the Largest Studied N

run/HPC_DEMO/img/N200_jobXXXX_task200_best_N200.pdf

Observations and Next Steps

Observed Behavior

- Small N: dense, visually near-optimal packings
- Large N: increasing slack, empty regions near boundary, disordered orientations

Immediate Next Steps

- **Post-processing (cheap):** greedy projection + shrink loop (boundary tightening)
- **Local refinement:** introduce smooth penalties and apply L-BFGS-B
- **Structure discovery:** periodic-boundary SA to find motifs, then tile into near-square patches

Summary

- Non-convex polygon packing solved with **two-phase SA** + robust geometry checks.
- **Area bound** provides provable infeasibility cutoff.
- **Bracket + bisection** yields a tight feasible L reliably.
- **Time-limited polish** improves L under strict HPC budgets and guarantees artifacts.
- Sweep diagnostics quantify **density**, **boundary waste**, and **structural disorder**.

Next steps:

- periodic-boundary runs to estimate infinite-tiling density,
- template/warm-start based on discovered motifs,
- targeted long-polish on hard N values.

Questions?