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

Bracketing, Bisection Refinement, and Time-Limited Shrink Polish

Joel Amir D. Maldonado Tänori

Applied Mathematics PhD, University of Arizona

January 15, 2026

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
6. HPC reliability: determinism, checkpoints, SIGTERM flush
7. Results / 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 $i = 1, \ldots, N$):

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

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 closer to *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, differentiable enough for SA, and effective when combined with increasing penalty weights in Phase B.

Energy Function and Feasibility Metric

We separate the concept of **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 (within an SA run 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 $\mathrm{feas} \leq \tau$ 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:** $(c_x, c_y) \leftarrow (c_x, c_y) + \Delta$ with $\Delta \sim \mathrm{Unif}([-s, s]^2)$.
- **Rotation mix:** with probability p_{rot}, $\theta \leftarrow \theta + \Delta\theta$.

Incremental bookkeeping. Only terms involving instance k need recomputation:

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

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

Phase B (Enforce).

- Lower temperatures
- Smaller step sizes
- **Ramping** penalties every K iterations: $(\lambda, \mu) \leftarrow \min((\lambda, \mu) \cdot \rho,\ (\lambda_{\text{max}}, \mu_{\text{max}}))$
- Purpose: aggressively drive $\mathrm{feas} \to 0$

SA Acceptance Rule

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

$$\Delta\mathcal{E} = \mathcal{E}' - \mathcal{E}.$$

$$\mathbb{P}(\text{accept}) = \begin{cases} 1, & \Delta\mathcal{E} \leq 0, \\ \exp(-\Delta\mathcal{E}/T), & \Delta\mathcal{E} > 0. \end{cases}$$

Key detail. We track the **best feasibility** configuration seen in the run:

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

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

Pseudocode: SA Trial at Fixed L

linewidth **SA_Trial(L)**

Initialize state x (positions + angles), compute to

Outer Loop Overview

We want the smallest feasible L. The outer optimizer proceeds as:

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

Reliability constraints. Area bound prevents invalid brackets; best feasible configuration is carried across L updates via scaling.

Bracketing Logic (with Area Bound)

We maintain:

$$L_{\text{low}} \text{ (infeasible)}, \qquad L_{\text{high}} \text{ (feasible)}.$$

Case 1: initial feasible. Repeatedly try $L \leftarrow \alpha L$ with $\alpha < 1$:

- Stop if $L \leq L_{\text{area-infeas}}$ (provably infeasible).
- Otherwise run SA at new L.
- If infeasible, set $L_{\text{low}} \leftarrow L$.

Case 2: initial infeasible. Repeatedly try $L \leftarrow \beta L$ with $\beta > 1$ until feasible; set L_{high}.

Important detail. We do *not* overwrite an SA-found infeasible lower bracket with the weaker area bound; we keep the tighter information.

Bisection Refinement

Given a valid bracket $L_{\text{low}} < L_{\text{high}}$:

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

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

Else:

- Warm-start by scaling best-feasible configuration from L_{high} to L_{mid}.
- Run a bounded number of SA trials at L_{mid}.
- If feasible: $L_{\text{high}} \leftarrow L_{\text{mid}}$ (and update best-feasible snapshot).
- Else: $L_{\text{low}} \leftarrow L_{\text{mid}}$ and restore best-feasible state.

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

Pseudocode: Bracket + Bisection

linewidth **Minimize L (Bracket + Bisection)**

Initialize L from grid lay- out:

Polish Stage: Adaptive Shrink Search

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

Idea. Repeatedly attempt a small shrink:

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

warm-start by scaling positions, then run a few SA trials.

If feasible: accept and update best solution.

If infeasible: reject and adjust ϵ.

Adaptive control. Maintain a sliding window of attempts and tune ϵ toward a target success rate (stochastic descent with backoff).

Time Limit and Checkpoints

For HPC sweeps, polish is time-limited (e.g., 900s = 15 minutes) to guarantee completion.

Periodic checkpoints (every Δt seconds):

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

SIGTERM handling. If Slurm sends SIGTERM near walltime:

- Flush **best snapshot** to both best and checkpoint files.
- Exit cleanly.

Outcome. Even canceled jobs yield usable artifacts.

HPC Engineering: Determinism and File Safety

Deterministic seeds. Each trial seed derived from:

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

(using SplitMix64 diffusion), ensuring reproducibility across arrays.

Unique output prefixes. For SLURM arrays:

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

Prevents file collisions when tasks run concurrently.

Compile-on-node option. Avoid GLIBC mismatch by building on compute nodes (cluster dependent).

Complexity Notes and Practical Performance

Theoretical worst case. Collision evaluation can be heavy, but practical performance is improved by:

- Uniform grid neighbor enumeration (local pairs)
- AABB + bounding-circle rejects (broad phase)
- Small fixed triangulation size (N_T constant)
- Incremental energy updates per move (only one instance changes)

Scaling expectation. Runtime grows with N mainly due to increased local density and neighbor interactions, not purely N^2.

Result Artifacts Produced Per Task

For each N, the code writes:

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

The CSV header includes:

- prefix, run_id, seed
- L and best feasibility score
- N

Recommendation. For figures, use SVG exports directly in LaTeX (or convert to PDF for best typography).

Example Best Solutions (Insert Your Figures)

img/N10_jobXXXX_task10_best_N010.pdf

img/N50_jobXXXX_task50_best_N050.pdf

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 further under strict HPC budgets.
- Engineering choices ensure **reproducibility** and **artifact safety** on Slurm arrays.

Next steps:

- Aggregate sweep results into an $L^{\star}(N)$ curve
- Compare against tiling-inspired warm-starts
- Increase SA trials for hard N and perform longer polish on selected cases

Questions?