A Shared-Codebase Ablation Study of GA, PSO, and ACO for Grid-Based Robot Path Planning

Junior Charlie

***Abstract*—Grid-based robot path planning must balance two objectives: a short path and a smooth one. A mobile robot pays a time and energy cost at every heading change, a cost a pure shortest-path solver ignores. This paper compares a genetic algorithm (GA), particle swarm optimization (PSO), and ant colony optimization (ACO) against an A* baseline, using one shared cell-sequence representation, fitness function, and A*-based repair routine across all three. We benchmark the four methods on nine grid instances (10×10, 20×20, and 50×50, at obstacle rates from 10% to 50%) over 30 seeds each, reporting path length, turn count, and Wilcoxon signed-rank significance against A* on turns. All three EAs match A*'s optimal length in eight of nine instances: turn reduction, not length reduction, is where they earn their keep. GA lowers mean turns significantly against A* in eight of nine instances (cuts of 4.4–11.9% where significant); PSO reaches significance in all nine; ACO in only two, tying rather than losing to A* on the two densest large-grid conditions. A follow-up ablation on the same shared codebase isolates which component actually drives GA's advantage: top-5% elitism changes too few seeds per instance to test at all, A*-seeded initialization reaches significance in only one of nine instances, and a stagnation-triggered mutation boost combined with population reinjection, proposed in the original design but never implemented, is by far the strongest lever, significant in seven of nine instances; splitting that combined mechanism further shows reinjection, not the mutation-rate increase, is the active ingredient. The same reinjection idea, transplanted onto ACO's ant-construction step, has no measurable effect on the two instances where ACO underperforms, and shows a benefit only on the small, sparse grids where ACO was already winning; a route-diversity argument explains why. We also document where the comparison breaks down: ACO occasionally returns a longer-than-optimal path on the densest small grid, and grid connectivity itself collapses under density, leaving only 11 of 30 seeds solvable on the densest 50×50 map. Runtime scales sharply with grid size for GA and PSO, both of which route every offspring through the same A*-based repair step, while ACO stays cheapest by a wide margin.**

***Index Terms*—path planning, genetic algorithm, particle swarm optimization, ant colony optimization, grid map, evolutionary computation, ablation study, A*, Wilcoxon signed-rank test**

I. Introduction

A mobile robot moving through a grid-mapped environment fails in two different ways. It can take a path that is too long, or it can take a path that turns too often. Classical graph search handles the first failure well: A* and its relatives return a length-optimal path whenever one exists. They do nothing about the second. When several paths tie on length, a priority-queue search returns whichever one it happens to expand first, with no preference for a straight trajectory over a jagged one. Every one of those extra turns costs a real robot time, energy, and a fresh round of sensor reacquisition, none of which shows up in a pure shortest-path objective.

This tradeoff between path length and turn count is the problem this paper studies. We take three evolutionary algorithms built for exactly this kind of dual objective, a genetic algorithm (GA), particle swarm optimization (PSO), and ant colony optimization (ACO), and run them against an A* baseline on the same nine grid configurations, from a sparse 10×10 map to a 50%-obstacle 50×50 map, over 30 random seeds apiece. Unlike most comparative studies in this space, all three metaheuristics here share one codebase: the same cell-sequence chromosome, the same fitness function, the same A*-seeded initialization, and the same simplify-and-repair pipeline for turning an invalid candidate path back into a valid one. That shared machinery removes a common confound in EA comparisons, where each algorithm brings its own encoding and its own repair logic, and differences in performance can come from implementation detail rather than search strategy.

A. Motivation and Problem Statement

Path length and turn count pull in different directions. The straight-line-adjacent paths that minimize length often weave between obstacles in ways that maximize direction changes, while a path that favors long straight runs to cut turns will usually give up some length to get them. A grid environment sharpens this conflict: with only eight allowed headings per cell, a planner threading a dense obstacle field is forced into frequent heading changes just to stay on the shortest route, and obstacle density itself changes the shape of the problem as it rises. Section 4 formalizes this with a scalar fitness function that weights length, turns, and a hard collision penalty, and Section 7 shows in the raw data how far a length-only planner drifts from a turn-aware one as that density increases.

B. Contribution

This paper offers four contributions.

1. A controlled, shared-implementation comparison of GA, PSO, and ACO for grid-based dual-objective path planning, where all three algorithms search over the same representation and reuse the same fitness, repair, and initialization code, isolating search strategy as the only free variable.

2. A benchmark across nine grid size / obstacle density combinations with 30 seeds each (270 runs per algorithm, 810 optimization runs total), reporting path length, turn count, and Wilcoxon signed-rank significance against the A* baseline on turns, together with wall-clock runtime for all three methods.

3. An evidence-based account of where each algorithm's assumptions break: ACO's occasional failure to preserve A*'s optimal length under high density in small grids, the collapse of grid connectivity itself at high obstacle rates, and the runtime cost that the shared A*-repair step imposes on GA and PSO as grid size grows.

4. A controlled ablation, run on the same shared codebase used for the main comparison, that attributes GA's turn-count advantage to its specific components rather than treating GA as a whole as the explanation, and that tests whether an anti-stagnation mechanism designed for GA transfers to ACO when transplanted onto its construction step.

C. Paper Organization

Section 2 reviews seven papers spanning heuristic GA initialization, potential-field PSO, Bézier-curve encoding, a WOA-PSO hybrid, two survey papers, and a cell-grid GA for UAV swarms. Section 3 formalizes the grid model, the dual objective, and the fitness function. Section 4 describes the shared path representation, the genetic operators, population initialization, and the discrete adaptations that turn PSO and ACO into grid-path searchers, closes with a structural comparison of all three algorithms, and details the design of a controlled ablation study built on top of that shared machinery. Section 5 lays out the experimental matrix, hyperparameters, and the rationale for a Wilcoxon signed-rank test over a paired t-test. Section 6 reports the results, with all ten figures and both results tables discussed against the raw data, presents the ablation study that isolates which specific mechanism drives GA's turn-count advantage, and tests whether the strongest of those mechanisms transfers to ACO's known failure modes. Section 7 concludes with the comparison's implications, its limitations, and directions for future work.

II. Related Work

Zhu and Pan [1] target the same weak point in classical GA path planning that motivates the initialization strategy used in this study: random population initialization on a grid map wastes most of its individuals on paths that never reach the goal or that collide with obstacles. Their fix is a population initialization method with directional guidance, which biases each individual's construction toward the goal cell from the start, paired with a non-common-point crossover operator and a range mutation operator that do not require two parents to share a fixed-position gene. Across four test maps, their directionally guided initialization raised the fraction of collision-free initial individuals well above two baseline initialization methods, and the resulting improved GA found the shortest path among five competing algorithms on every map tested. The GA implementation in this study borrows the same idea in a different form: rather than biasing random construction toward the goal, it seeds the entire initial population from a single A* solution and diversifies it by repeated perturbation, which sidesteps the collision-prone-individual problem entirely rather than reducing its frequency.

Zheng et al. [2] address a different weak point in PSO-based path planning: the classical particle swarm algorithm converges toward a local optimum in obstacle-dense environments and its search direction offers no explicit obstacle awareness. They introduce apfrPSO, which first reorders the particle position vectors and adjusts the inertia weight to balance global and local search, then layers an artificial potential field (APF) on top so that particle velocity updates are pulled by target attraction and pushed by obstacle repulsion simultaneously. The APF term is what gives their method faster, more accurate convergence than plain PSO in their reported comparisons. The PSO variant benchmarked here takes a structurally different route to the same discrete search problem: velocity is redefined as crossover toward the personal-best and global-best paths rather than as a continuous vector update, so there is no APF-style repulsion term, and obstacle avoidance instead falls entirely on the shared simplify-and-repair pipeline described in Section 4.

Ma et al. [3] take encoding, not initialization or velocity, as the point of leverage. Their GA works over discrete waypoints on a grid, but the fitness of each candidate is evaluated after fitting a continuous Bézier curve through its control points, with a safety-distance term and an adaptive penalty factor added to the fitness function to keep the curve clear of obstacles. Because a Bezier interpolation removes sharp corners by construction, their method reports shorter and smoother output paths than comparable discrete-waypoint GAs across two test environments. The comparison in this paper stays entirely in the discrete cell-sequence domain: turns are counted directly on the grid path rather than measured as curvature along an interpolated curve, which keeps the fitness function in Section 3 simple but forfeits the smoothing Ma et al. get for free from the curve fit.

Najm et al. [4] hybridize two metaheuristics rather than modifying one: their HWPSO pairs the whale optimization algorithm's global exploration with PSO's local exploitation, switching between the two search behaviors according to a control parameter, and applies the result to differential wheeled mobile robots. Relative to competing methods in their own benchmark, HWPSO cuts path length by close to a fifth and travel duration by roughly an eighth. Their result is a useful reminder that a hybrid search strategy, not just a better encoding or fitness function, is itself a lever on solution quality, though hybridization sits outside the scope of the single-strategy comparison run here, where GA, PSO, and ACO are each evaluated in their own right rather than combined.

Tang et al. [5] survey the broader landscape this paper's three algorithms sit inside, tracing the field's move from classical planners such as A* and Dijkstra toward intelligent, learning-based, and hybrid approaches, and reporting that fused multi-algorithm methods consistently outperform single-algorithm path planners in the studies they review. That survey-level finding is a natural prediction to test empirically, and Section 6 does exactly that with GA, PSO, and ACO run individually rather than fused, providing a same-representation, same-fitness data point against which any

future hybrid built on this codebase could be measured.

Qin et al. [6] organize mobile robot path planning into eight algorithm families and give particular attention to hierarchical combinations of a global planner with a local, learning-based refinement stage. Among the approaches they summarize is a two-stage design that plans a static global path with a genetic algorithm offline, then hands dynamic obstacle avoidance to a Q-learning policy online, a division of labor between an evolutionary global planner and a reinforcement-learned local controller. The comparison here stays entirely in the static, fully-known-map setting; none of GA, PSO, or ACO in this study receives an online learning component, so the dynamic-obstacle capability Qin et al. describe is a direction for extension rather than a feature under test.

Puente-Castro et al. [7] apply a genetic algorithm to full-coverage path planning for UAV swarms on cell-grid maps, and along the way flag a specific encoding hazard in prior grid-based GA work: when a path is allowed to pass through grid intersection points rather than being confined to cell centers, identifying which discrete cell a given waypoint belongs to becomes ambiguous. Their own encoding sidesteps the problem by tying each gene to one movement direction per cell rather than to a raw coordinate. The grid representation in this study avoids the same hazard by construction: every waypoint is a cell index from an N×N integer grid, so no waypoint can ever fall between cells, and the ambiguity Puente-Castro et al. flag does not arise here. Their proposed extensions, letting a UAV revisit already-covered cells and giving each vehicle its own movement schedule, foreshadow the multi-robot direction discussed as future work in Section 7.

Across these seven studies, the common move is to propose a new operator or hybrid, a directional initializer, an APF-augmented velocity update, a Bézier-smoothed fitness, a whale-PSO hybrid, and evaluate it against a small set of baselines. None ablates an existing design to ask which of its own components is doing the work, a methodological gap the present study addresses directly, following a broader argument in the heuristics literature for controlled experimentation over competitive testing [8]: rather than adding a new mechanism and measuring whether the aggregate result improves, Section 4.7 and Section 6.4 remove one mechanism at a time from an already-benchmarked design and measure what changes.

III. Problem Formulation

The environment is an N × N binary occupancy grid $G \in \{0,1\}^{N\times N}$, where $G[r,c]=1$ marks cell (r,c) as an obstacle and $G[r,c]=0$ marks it free. The start cell is fixed at $(0,0)$ and the goal at $(N-1,N-1)$; both are forced free regardless of the sampled obstacle rate. A robot occupying free cell (r,c) may step to any of the eight Moore-neighborhood cells $(r+\delta r, c+\delta c)$, $\delta r, \delta c \in \{-1,0,1\} \setminus \{(0,0)\}$, that lie inside the grid and are themselves free. An orthogonal step costs 1 and a diagonal step costs $\sqrt{2}$, so the cost of moving from cell a to an adjacent cell b is the Euclidean distance between their grid coordinates,

$$\mathrm{cost}(a,b) = \sqrt{(r_a - r_b)^2 + (c_a - c_b)^2}. \tag{1}$$

A path $\pi = (\pi_0, \pi_1, \ldots, \pi_k)$ is a sequence of cell indices with π_0 the start, π_k the goal, and π_{i+1} an 8-connected free neighbor of π_i for every i.

Two objectives are in tension over the space of valid paths. The first is path length,

$$L(\pi) = \sum_{i=0}^{k-1} \mathrm{cost}(\pi_i, \pi_{i+1}), \tag{2}$$

which A* minimizes exactly under this cost model. The second is the turn count,

$$T(\pi) = \big|\{\, i \in [1, k-1] : \Delta_{i-1,i} \neq \Delta_{i,i+1} \,\}\big|, \tag{3}$$

where $\Delta_{i,j}$ is the discrete heading $(r_j - r_i, c_j - c_i)$ of the step from π_i to π_j. $T(\pi)$ counts every interior waypoint at which the heading changes, with no distinction between a 45° turn and a 135° turn. A* has no mechanism to prefer one length-optimal path over another on this second objective: whichever length-optimal path its priority queue happens to pop first is the one it returns, so two runs that differ only in tie-breaking order over an unordered structure can return paths with very different turn counts for the identical grid.

To search over both objectives with a single scalar signal, every candidate path is scored by a weighted-sum fitness function,

$$f(\pi) = w_1 L(\pi) + w_2 T(\pi) + w_3 C(\pi), \tag{4}$$

with $w_1 = 0.4$, $w_2 = 0.2$, $w_3 = 100.0$, and $C(\pi)$ the count of obstacle cells the path passes through. All three algorithms in this study minimize Equation 4 directly. The w_3 term is a hard collision penalty rather than a soft preference: at $w_3 = 100$, a single obstacle cell costs more fitness than dozens of extra turns or a substantially longer detour, which keeps the search pinned to collision-free paths whenever one is reachable rather than trading a small amount of collision against a large amount of smoothness. w_1 outweighing w_2 two-to-one reflects the same design choice as the underlying A* baseline: length still dominates the objective, and turn count acts as a tie-breaker among near-length-optimal paths rather than a co-equal objective that could pull the search toward a much longer but perfectly straight route.

Obstacle density interacts with this formulation in a way that a pure length objective conceals. As the fraction of blocked cells rises, the number of length-optimal paths through the surviving free cells falls, and each of those survivors is forced through narrower gaps where every reasonable route must turn to avoid an obstacle immediately ahead. Below a critical density the grid remains connected end to end and A* returns a solution regardless of how jagged it is; above it, connectivity between start and goal breaks down entirely and no path, planned by any algorithm, exists. Section 6 shows this second effect directly: on the 50×50 grid at 50% obstacle density, 19 of 30 randomly sampled instances have no start-to-goal path at all, a regime where the dual-objective comparison this paper runs is only defined for the minority of instances that remain solvable.

IV. Method

A. Path Representation

All three algorithms search over the same chromosome: a variable-length list of grid cell indices from start to goal, each consecutive pair an 8-connected free-cell edge. This choice is deliberate. A fixed-length or continuous encoding, such as the Bézier control points used by Ma et al. [3], needs a separate decoding step before a path can be checked for collisions or measured for turns, and that decoding step has to be re-derived for every algorithm that reuses it. A raw cell-sequence chromosome is already a path: $L(\pi)$, $T(\pi)$, and $C(\pi)$ from Section 3 read off it directly, and any two cell-sequence chromosomes, regardless of which algorithm produced them, can be repaired, simplified, or recombined with the identical routines. That interchangeability is what lets this study's GA, PSO, and ACO implementations share one fitness function, one repair routine, and one initializer rather than three separate ones, and it is also what leaves the grid-intersection ambiguity flagged by Puente-Castro et al. [7] without a foothold: every gene is an integer cell index on an N×N grid, never a coordinate that could fall between cells. Cell-sequence was one of three encodings weighed at the design stage, alongside a direction-sequence chromosome, where each gene is a heading rather than a cell and the walk is connected by construction but needs a bounded length limit, and a waypoint chromosome interpolated into a continuous curve, of the kind Section 4's Bézier comparison already touches on. Cell-sequence was chosen for the reason above: every intermediate chromosome, valid or not, is directly visible and directly measurable, which matters as much for debugging during development as for the shared-tooling argument this section makes.

B. Population Initialization

Every population, whether a GA population, a PSO swarm, or the pheromone field an ACO colony walks over, starts from the same A* solution. The initializer runs A* once to get a baseline path, then builds the remaining population by repeatedly copying that baseline, applying one to four random perturbations (Section 4.3), and repairing the result back to validity. If repeated perturbation fails to produce enough distinct individuals, the population is padded with unperturbed copies of the baseline. This is a stronger commitment to a warm start than the directionally guided random construction of Zhu and Pan [1], whose heuristic-median-insertion initializer routes each individual's random construction through the midpoint between start and goal rather than seeding directly from A*, and which the framework document behind this project credits with a 30–60% convergence-time reduction on 50×50 grids, a figure that belongs to their reported experiments, not to the runs in this study. Rather than biasing a random build process toward the goal, every individual in every run of this study begins life as a small variation on a path that is already known to be length-optimal, and the entire subsequent search only has to find turn-reducing variations on that baseline rather than rediscover a feasible route from scratch. ACO deviates from this scheme only in how it uses the baseline: instead of seeding a population, it deposits pheromone along the A* path before the first ant ever walks, a warm start that shortens the cold-start phase during which a uniform pheromone field would otherwise send ants wandering with no directional bias at all.

C. Genetic Operators

The GA advances a population of size 50–120 (Section 5) for 300–800 generations using four operators layered on top of the shared representation.

Selection. Tournament selection picks the fitter of $k = 3$ randomly drawn individuals; the top 5% by fitness are carried into the next generation unchanged (elitism), guaranteeing the population's best fitness never regresses across a generation.

Crossover. The non-common-point operator, applied with probability $p_c = 0.8$, finds the set of interior cells shared by both parents, picks one at random, and splices the first parent's prefix up to its first occurrence of that cell onto the second parent's suffix from its last occurrence of the same cell onward. Taking the second parent's suffix from its last occurrence, rather than its first, guarantees the child still ends at the goal even when the shared cell appears more than once in either parent. If the two parents share no interior cell, the operator returns the first parent unchanged rather than failing.

Mutation. The range-mutation operator, applied with probability $p_m = 0.05$, makes a single local edit: with equal probability it either inserts a new waypoint between two consecutive existing ones (found by intersecting their neighbor sets) or deletes an interior waypoint whose neighbors on either side are already adjacent. If a chosen edit type finds no candidate, it falls back to the other type once before giving up on that mutation call. This is a single-probe local search, not an exhaustive scan of every possible insertion or deletion point, which keeps the operator's cost independent of path length. The function signature still carries a `delta=2` parameter from an earlier design that perturbed a waypoint within a bounded ± 2-cell neighborhood; the implemented operator no longer reads it, having moved to the shared-neighbor insert/delete scheme described above, so the parameter is dead code rather than an active tuning knob.

Simplification and repair. Every offspring, after crossover and mutation, passes through two cleanup steps before it re-enters the population. Simplification walks the path and drops any interior waypoint whose incoming and outgoing headings are identical, which removes turns that crossover or mutation left behind as a side effect, at zero fitness cost, since a collinear waypoint changes neither the endpoints nor the set of cells visited. Repair then checks the result for validity and, for any broken edge, calls A* locally to reroute from the last valid waypoint to the next surviving one, falling back to the original unrepaired child only if even that local reroute fails, in which case the fitness function's collision term $C(\pi)$ is left to penalize it.

D. Discrete Adaptation: PSO

Continuous PSO updates each particle's position by adding a velocity built from three pulls: inertia toward the particle's current direction of travel, a cognitive pull toward its own best-known position, and a social pull toward the swarm's best-known position. None of those three pulls has a natural meaning for a variable-length list of grid cells, so this study's PSO redefines velocity as a sequence of crossover and mutation events applied directly to the particle's current path. At every iteration, a particle first crosses over toward its personal best via `crossover_ncp`, whenever a random draw is under the cognitive weight c_1, then crosses over toward the swarm's global best whenever a second random draw is under the social weight c_2, and finally applies the same `perturb` mutation used by the GA whenever a third draw is under the inertia weight w. One implementation detail is worth stating plainly: with $c_1 = c_2 = 1.5$, both draws against c_1 and c_2 are checked against a uniform $[0, 1)$ random number, so both conditions hold on every particle on every iteration. In this codebase, c_1 and c_2 do not function as selective probabilities the way values above 1 would suggest; every particle crosses over toward both its personal best and the global best on every single iteration, and it is the inertia weight $w = 0.5$, used here as a genuine mutation probability rather than a blend coefficient, that is the only one of the three classical PSO parameters retaining a probabilistic role in this implementation. Every particle's result is simplified, repaired, and re-scored exactly as a GA offspring would be, and personal-best and global-best bookkeeping is updated whenever the new fitness improves on the stored one.

E. Discrete Adaptation: ACO

ACO replaces the population entirely with a per-cell pheromone field $\tau \in \mathbb{R}^{N \times N}$, initialized uniformly and then boosted along the warm-start A* path described in Section 4.2. Each of n_{ants} ants per iteration constructs a path by stepping, from the start cell, to an unvisited free neighbor chosen by roulette-wheel sampling with weight $\tau_c^{\alpha} \cdot \eta_c^{\beta}$ for each candidate neighbor c, where η_c is a precomputed heuristic equal to the inverse Chebyshev distance from c to the goal. An ant that reaches a cell with no unvisited free neighbor left, and has not yet reached the goal, is discarded for that iteration rather than backtracked; a hard step cap of $2N^2$ guards against pathological wandering. This design departs from a textbook edge-pheromone ACO in two ways that trade some search fidelity for a smaller memory footprint and a simpler update rule: pheromone lives on cells rather than on the $8N^2$ directed edges of the grid graph, and there is no 2-opt or other local-search daemon action layered on top of the ant-colony construction step. After every iteration, all successful ant paths are simplified, pheromone evaporates uniformly by a factor $(1 - \rho)$ and is clipped to a small positive floor to keep every cell selectable, and each successful path deposits $Q/L(\pi)$ pheromone along the cells it visits, so shorter paths reinforce their route more strongly than longer ones.

F. Structural Comparison

Table 1 summarizes how the three algorithms differ once every representation-level decision is held fixed. GA and PSO diverge only in how they turn the shared crossover and mutation primitives into a per-iteration update rule; ACO discards those primitives in favor of pheromone-guided stepwise construction and never calls repair, since its ant-walk construction is restricted to free, unvisited neighbors by definition and therefore cannot produce a colliding path in the first place.

G. Ablation Design

Sections 4.1–4.6 describe only the fully-assembled "full" configuration of each algorithm benchmarked in Section 6. To find out which of those design choices is actually responsible for GA's turn-count advantage, and whether any lesson from that analysis carries over to ACO, this study runs a second experimental matrix: the same nine grid/obstacle-rate instances and the same shared fitness function, repair pipeline, and A* baseline, but with one mechanism at a time removed from GA or added to ACO. This follows a broader argument in the heuristics literature for controlled experimentation over competitive testing [8]: instead of comparing three complete, differently-implemented algorithms and inferring why one wins, the ablation holds the implementation fixed and varies one component, so any resulting difference has a single, attributable cause.

Six GA configurations are compared, each run for the same seed budget as the corresponding grid size in Section 5.2 at 10×10 and 20×20 (30 seeds), and a reduced 10-seed subsample (seeds 0–9) at 50×50, for runtime reasons:

- `full`: the configuration benchmarked in Section 6, unchanged.
- `no_astar_init`: swaps the A*-seeded population of Section 4.2 for a directionally-biased random walk toward the goal, followed by the same repair step; no A* call occurs anywhere in the seeding step itself.
- `no_elitism`: drops the top-5% elite carve-forward described in Section 4.3; tournament selection is otherwise unchanged.
- `adaptive_mut`: boosts the mutation probability p_m and reinjects a fresh, randomly re-perturbed 10% worst-fraction of the population once a stagnation window (a fixed number of generations with no improvement to the best fitness) is reached, then relaxes both once fitness improves again. This is the mechanism this project's original design framework specified as a defense against premature convergence (Section 6.7) but that the GA benchmarked in Section 6 does not implement.
- `adaptive_pm_only` and `reinject_only`: `adaptive_mut` bundles two separate mechanisms, a mutation-rate increase and a population-reinjection step, triggered by the same stagnation signal. Because a single combined ablation arm cannot say which of the two is responsible for any effect it produces, `adaptive_mut` is split into these two configurations, each toggling only

TABLE I
STRUCTURAL COMPARISON OF THE THREE EVOLUTIONARY ALGORITHMS, ALL BUILT ON THE SHARED REPRESENTATION AND CODE OF SECTIONS 4.1–4.2.

	GA	**PSO**	**ACO**
Search unit	Chromosome	Particle	Ant walk
Guidance	Tournament + elitism (top 5%)	pbest/gbest crossover	Pheromone $\tau^{\alpha}\eta^{\beta}$
Variation	`crossover_ncp` (p_c=0.8), `perturb` (p_m=0.05)	`crossover_ncp` ×2 (always), `perturb` (w=0.5)	Roulette-wheel step
Validity	`simplify+repair` every child	`simplify+repair` every particle	Free-neighbor-only walk; `simplify` only
Memory	Elite carried forward	Per-particle pbest, single gbest	Evaporating pheromone field
Init.	A*-seeded, perturbed	Shared with GA	Uniform + A* warm-start deposit
Size used	50–120 (scales with N)	80 (fixed)	50 ants (fixed)
Iterations	300–800 (scales with N)	200 (fixed)	200 (fixed)

one of the two mechanisms. The mutation-rate-boost side of this pairing is the same lever explored as triggered hypermutation in the dynamic-optimization literature, and the reinjection side is a fixed-fraction variant of the random-immigrants scheme from the same literature [9]; both were originally proposed to help a GA track a moving optimum, not, as here, to escape stagnation on a fixed one, but the underlying diversity-restoration mechanism is the same.

One ACO configuration, `restart_dead_ends`, is compared against the ACO baseline already reported in Table 3: instead of discarding an ant that reaches a cell with no unvisited free neighbor (Section 4.5), the ant's walk is replaced with a perturbed-and-repaired copy of the colony's current best path, so a dead end costs a wasted construction step rather than a lost ant. This mirrors, in miniature, the stagnation-recovery role that pheromone-trail reinitialization plays in MAX–MIN Ant System [10], though `restart_dead_ends` acts on individual dead-ended walks within an iteration rather than on the shared pheromone field between iterations. Diversity-injection mechanisms have been added to ACO before, but for a different purpose: Mavrovouniotis and Yang integrate random- and elitism-based immigrants schemes into ACO so it can track a moving optimum on dynamic routing problems [11], finding elitism-based immigrants preferable when the environment changes slowly. The grids here never change once sampled, so `restart_dead_ends` is closer in spirit to their elitism-based variant, rebuilding from the current best rather than a random walk, but it targets a different failure mode: escaping a structural dead end within one static search rather than adapting across changing ones.

Every ablation configuration is compared against its own reference point, `full` for the five GA variants and `baseline` for `restart_dead_ends`, with a Wilcoxon signed-rank test paired by seed, identically to the A*-comparison methodology of Section 5.3. Table 4's `full` column and Table 3's GA row draw on the same underlying `full`-configuration runs at 10×10 and 20×20, where both use the complete 30-seed sample, and the two columns are identical. At 50×50, Table 4 reports `full` over the same reduced 10-seed subsample used for the other five ablation arms, so that every column in Table 4 is compared against identical paired seeds, while Table 3's 50×50 row uses the complete 30-seed sample; the two tables' 50×50 numbers therefore differ slightly by sample size rather than by any difference in code or configuration.

V. EXPERIMENTAL SETUP

A. Benchmark Matrix

The benchmark spans three grid sizes, each tested at three obstacle rates: 10 × 10 at 10%, 20%, and 30%; 20 × 20 at 10%, 20%, and 40%; and 50 × 50 at 10%, 30%, and 50%. Each of the nine grid/rate combinations is instantiated with 30 independent random seeds (seeds 0–29), for 270 grid instances total. Every grid cell, aside from the start at $(0, 0)$ and the goal at $(N-1, N-1)$, is marked an obstacle independently with probability equal to the target rate, using a seeded NumPy generator so that every algorithm is tested against the identical set of grids for a given instance and seed. A* is run first on every instance to establish both the length-optimal baseline and a reachability check; instances where A* finds no path are excluded from that seed's comparison, since none of GA, PSO, or ACO is seeded or evaluated on an instance its own baseline could not solve. Because Section 3 already establishes that two independent A* runs on the same grid can land on different, equally-optimal paths whenever the shortest path is not unique, every reported A* statistic in Table 2 and Table 3, and every A*-vs-EA paired comparison, is computed against a single canonical A* solve per (instance, seed), generated once and reused across the GA, PSO, and ACO evaluation scripts rather than recomputed independently by each; this is the same A* baseline every other method in this study is initialized from (Section 4.2) and compared against.

B. Hyperparameters

GA hyperparameters scale with grid size, since a larger grid needs a larger population and more generations to explore a proportionally larger search space: population size is 50, 80, and 120 for the 10 × 10, 20 × 20, and 50×50 grids respectively, run for 300, 500, and 800 generations. Crossover probability $p_c = 0.8$, mutation probability $p_m = 0.05$, tournament size $k = 3$, and elitism at the top 5% are held fixed across all three grid sizes. PSO and ACO hyperparameters, by contrast, are held fixed across all nine instances: PSO runs 80 particles for 200 iterations with inertia $w = 0.5$ and

cognitive/social weights $c_1 = c_2 = 1.5$; ACO runs 50 ants for 200 iterations with pheromone weight $\alpha = 1.0$, heuristic weight $\beta = 2.0$, evaporation rate $\rho = 0.1$, and deposit constant $Q = 1.0$. Every run is wall-clock timed independently, single-core, one seed at a time, from just before the algorithm call to just after it returns.

C. Statistical Testing

Each EA is compared against the A* baseline with a Wilcoxon signed-rank test, paired by random seed within each of the nine instances. A paired, rank-based test is the right tool for this comparison for two reasons specific to this data. First, turn count is a small non-negative integer, and at low obstacle density a large share of seed-level differences between an EA and A* are exactly zero, either because both found the same path or because both happened to find equally-jagged alternatives; a paired t-test assumes the differences are drawn from a continuous, approximately normal distribution, an assumption a pile of exact zeros and a handful of small integer differences does not satisfy. Second, the Wilcoxon signed-rank statistic depends only on the relative ranks of the non-zero absolute differences, not on their magnitude, which is appropriate when a difference of 1 turn and a difference of 8 turns should both count as evidence of a directional effect without letting the larger value dominate the test the way a mean-based statistic would. This choice follows established guidance for comparing evolutionary and swarm-intelligence algorithms with nonparametric, paired procedures rather than assuming normality [12]. This study's implementation ranks the non-zero paired differences, resolves ties by averaged rank, and computes a p-value from the normal approximation to the signed-rank distribution, which holds for $n \geq 5$ effective (non-zero) paired observations; instances with fewer than 5 non-zero differences report an undefined test, shown as a dash in Table 3. Path length is not tested this way: because every EA reaches A*'s exact optimal length in the overwhelming majority of seeds (Section 6), the seed-level length differences are zero for nearly every pair, and a signed-rank test over a column of zeros carries no information. Section 6.4's ablation study reuses the identical test, paired by seed, to compare each ablation configuration against its own `full` or `baseline` reference rather than against A*.

VI. Results and Discussion

A. Qualitative Path Comparison

Figures 1–3 show one representative GA solution at 20% obstacle density on each of the three grid sizes, with red markers on every waypoint where the path turns. The turn count rises from 2 on the 10 × 10 grid to 7 on the 20 × 20 grid and 12 on the 50 × 50 grid, roughly tracking the growth in path length (13.31, 28.04, and 71.64 grid units respectively) rather than growing with obstacle density alone, which is already visible qualitatively before Table 3 confirms it numerically: a longer path has more opportunities to need a turn, independent of how cluttered the grid is.

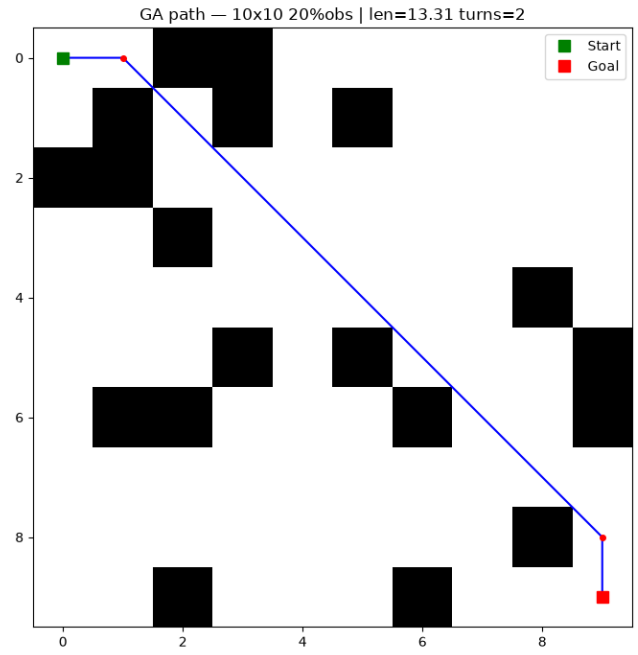

Fig. 1. GA-planned path on a 10 × 10 grid at 20% obstacle density (length 13.31, 2 turns). Red dots mark waypoints where the heading changes.

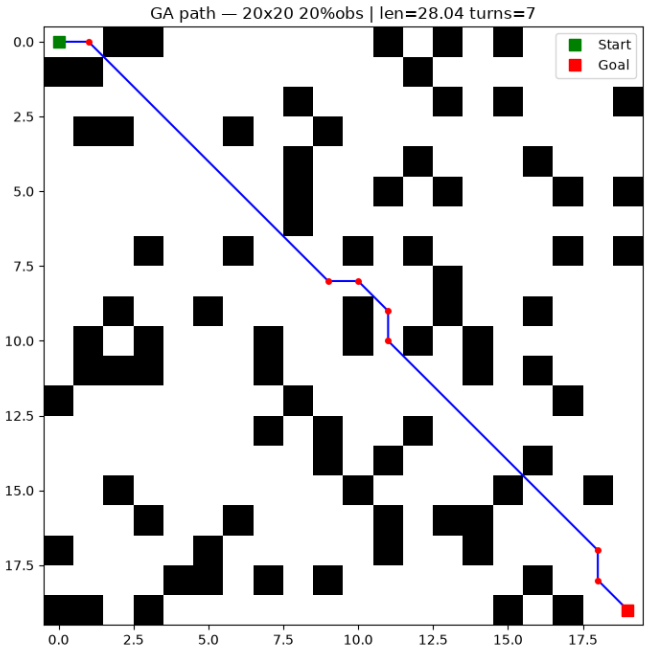

Fig. 2. GA-planned path on a 20 × 20 grid at 20% obstacle density (length 28.04, 7 turns).

B. Path Length

Table 2 reports mean and standard deviation path length for all four methods across all nine instances. The result is close to a flat line: GA and PSO match A*'s mean length to four decimal places in every one of the nine instances, and ACO matches it in eight of nine. A* is optimal on length

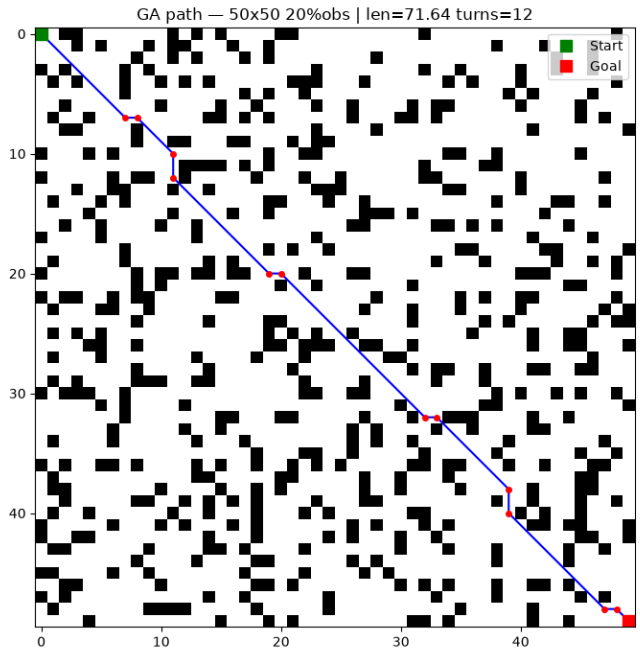

Fig. 3. GA-planned path on a 50 × 50 grid at 20% obstacle density (length 71.64, 12 turns).

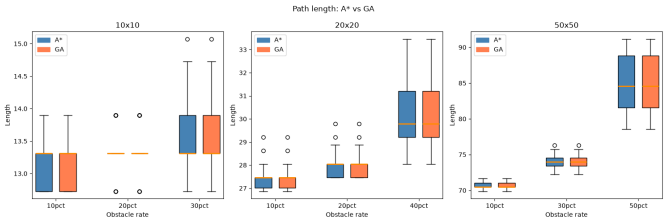

Fig. 4. Path length distributions, A* vs. GA, across all nine instances grouped by grid size and obstacle rate.

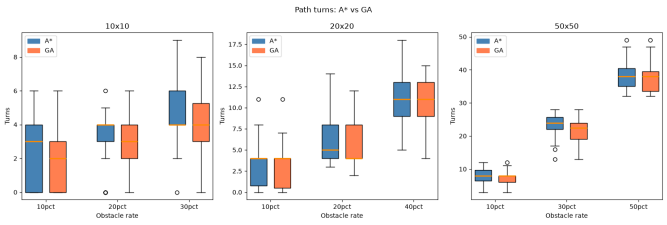

Fig. 5. Turn count distributions, A* vs. GA, across all nine instances.

by construction, so this outcome is exactly what a correctly implemented dual-objective EA should produce, provided its collision penalty and repair machinery are strong enough to keep the search from drifting toward a shorter-but-invalid or longer-but-smoother region of the search space purely to save on turns. Figures 4 and 6 show the same result visually: the A*-vs-GA boxplots and the four-way boxplots are visually indistinguishable within each grid/obstacle-rate group, which is the expected picture when three additional search algorithms all converge to the same length-optimal region A* already occupies.

The one exception is instructive. On the 10x10 grid at 30% obstacle density, ACO's mean length rises to 13.70 against 13.68 for the other three methods, a difference of about 0.15%, accompanied by a slightly larger standard deviation (0.54 vs. 0.52). This is the only one of the nine instances where any algorithm fails to match A*'s exact optimum on average, and it appears at the combination of the smallest grid and the highest obstacle rate tested at that size, exactly where an ant's stepwise, no-backtracking construction (Section 4.5) has the least room to route around a dead end: with few free cells left and no 2-opt or local-search correction applied afterward, an ant that reaches a corner with no unvisited free neighbor is simply discarded, and the walks that do survive to reach the goal are not guaranteed to be the length-shortest among the walks that could have been taken.

C. Turn Count and Statistical Significance

Turn count is where the three algorithms separate. Table 3 reports mean turns and the Wilcoxon p-value against A* for each EA, with bold entries marking $p < 0.05$. GA reaches significance in eight of the nine instances, cutting mean turns relative to A* by between 4.4% and 11.9% depending on the instance, with the largest cut at the sparsest condition tested on the 10x10 grid (10% obstacle density) rather than the densest; the sole exception is 50x50 at 50% obstacles, the smallest-sample condition in the benchmark ($n = 11$), where the reduction is not significant. PSO reaches significance in all nine instances, including the two large, dense conditions, 20x20 at 40% and 50x50 at 50%, that are the two hardest instances in the matrix. ACO reaches significance in only two of nine, both at the 10x10 grid (20% and 30% obstacle density); at the two instances where ACO might appear to underperform A*, 20x20 at 40% and 50x50 at 50%, the two produce identical mean turn counts, with zero non-zero paired differences at either instance, so ACO ties A* at its two weakest conditions rather than losing to it.

Figures 5 and 7 show this same ordering visually. In the pairwise A*-vs-GA comparison, GA's median turn count sits at or below A*'s median in every grid/rate group, most visibly on the 10 × 10 grid at 10% obstacle density, where GA's median drops to 2.0 turns against A*'s 3. In the four-way comparison, ACO's boxes track close to A*'s across most groups and, on the 50 × 50 panel in particular, sit visibly above both GA and PSO rather than below them, consistent with the corrected values in Table 3, where ACO's turn count equals A*'s at the two largest, densest conditions rather than falling below it.

An ablation isolating each candidate mechanism (Section 4.7, Table 4) clarifies which of these design choices actually drives GA's result; Section 6.4 reports it in full. In brief: removing the top-5% elitism carve-forward changes GA's turn count on too few seeds per instance to test in any of the nine conditions, so elitism is not the load-bearing mechanism a purely descriptive reading of Section 4.3 might suggest. Removing the A*-seeded initialization in favor of a directionally-biased random walk reaches significance in only

TABLE II

MEAN ± STANDARD DEVIATION PATH LENGTH (EUCLIDEAN GRID UNITS) FOR A*, GA, PSO, AND ACO ACROSS THE NINE BENCHMARK INSTANCES. n IS THE NUMBER OF SEEDS (OUT OF 30) FOR WHICH A START–GOAL PATH EXISTS.

Instance	n	**A***	**GA**	**PSO**	**ACO**
10x10, 10% obs	30	13.06±0.33	13.06±0.33	13.06±0.33	13.06±0.33
10x10, 20% obs	29	13.33±0.33	13.33±0.33	13.33±0.33	13.33±0.33
10x10, 30% obs	28	13.68±0.52	13.68±0.52	13.68±0.52	13.70±0.54
20x20, 10% obs	30	27.42±0.50	27.42±0.50	27.42±0.50	27.42±0.50
20x20, 20% obs	30	27.97±0.59	27.97±0.59	27.97±0.59	27.97±0.59
20x20, 40% obs	25	30.08±1.50	30.08±1.50	30.08±1.50	30.08±1.50
50x50, 10% obs	30	70.60±0.45	70.60±0.45	70.60±0.45	70.60±0.45
50x50, 30% obs	26	74.04±1.03	74.04±1.03	74.04±1.03	74.04±1.03
50x50, 50% obs	11	84.91±4.10	84.91±4.10	84.91±4.10	84.91±4.10

TABLE III

MEAN ± STANDARD DEVIATION TURN COUNT FOR A*, GA, PSO, AND ACO, WITH WILCOXON SIGNED-RANK P-VALUES (PAIRED BY SEED) FOR EACH EA AGAINST A* ON TURNS. BOLD ENTRIES DENOTE $p < 0.05$. A DASH INDICATES FEWER THAN 5 NON-ZERO PAIRED DIFFERENCES (TEST NOT MEANINGFUL).

Instance	**A***	**GA**	**PSO**	**ACO**	p_{GA}	p_{PSO}	p_{ACO}
10x10, 10% obs	1.97±1.92	1.73±1.77	1.70±1.73	1.83±1.81	**0.028**	**0.012**	—
10x10, 20% obs	3.14±1.48	2.83±1.39	2.90±1.40	2.90±1.40	**0.018**	**0.018**	**0.018**
10x10, 30% obs	4.71±1.79	4.25±1.70	3.93±1.36	4.14±1.57	**0.005**	**0.002**	**0.005**
20x20, 10% obs	3.23±2.43	3.07±2.37	3.00±2.35	3.23±2.43	**0.043**	**0.018**	—
20x20, 20% obs	6.07±2.78	5.70±2.54	5.40±2.33	6.07±2.78	**0.012**	**0.002**	—
20x20, 40% obs	10.96±3.38	10.48±3.26	10.32±3.32	10.96±3.38	**0.028**	**0.008**	—
50x50, 10% obs	7.97±2.54	7.30±2.18	7.43±2.32	7.97±2.54	**0.008**	**0.008**	—
50x50, 30% obs	23.00±3.57	21.85±3.89	22.00±3.93	23.00±3.57	**0.003**	**0.003**	—
50x50, 50% obs	38.73±5.17	38.00±5.41	37.27±5.26	38.73±5.17	—	**0.028**	—

one instance, the sparsest 10×10 condition, and even there the effect is small; on several other instances the mean moves in the opposite direction, though never significantly. What does move the needle, consistently and often significantly, is a mechanism absent from the GA benchmarked in this table altogether: reinjecting a fraction of the population after a stagnation window. A mutation-rate increase triggered by the same stagnation signal contributes comparatively little on its own, a distinction Section 6.4's finer-grained ablation makes explicit.

D. GA Ablation Study

Section 6.3's turn-count advantage is GA's headline empirical result, but Table 3 alone cannot say which part of GA's design, A*-seeded initialization, tournament selection with elitism, or some other mechanism, is responsible for it. Section 4.7 describes six GA configurations built to answer that question by removing or adding one mechanism at a time from the `full` configuration benchmarked in Section 6.3; Table 4 reports the result, each variant's mean turn count and a Wilcoxon signed-rank p-value paired by seed against `full`.

Two of the three originally-suspected mechanisms turn out not to explain the result. `no_elitism` produces fewer than five non-zero paired differences from `full` in every one of the nine instances, the Wilcoxon test's minimum sample size (Section 5.3), so the effect of removing elitism cannot be distinguished from noise in any tested condition; whatever the top-5% carve-forward contributes to preventing regression across generations, it is not measurably responsible for GA's turn-count advantage over A*. This is a theoretical-versus-empirical distinction, not a contradiction: elitist selection, keeping the best-so-far individual, is what gives a genetic algorithm a convergence guarantee at all [13], but a guarantee that fitness never regresses says nothing about how much of the observed turn-count gap it is responsible for on a fixed, short run, and this ablation's answer is: measurably little. `no_astar_init` clears the significance bar at a single instance out of nine, the sparsest 10x10 condition, and the direction is inconsistent across the other eight: the mean moves higher in some instances (worse without A* seeding) and lower in others (better without it), never significantly either way. Both results are consistent with Section 4.2's warm-start argument in a narrower sense than originally framed: A*-seeded initialization very likely still saves search effort relative to reaching a valid path from scratch, a claim this ablation does not test, but it is not the specific mechanism separating GA's final turn count from A*'s.

The mechanism that is responsible was, at the design stage, never implemented at all. `adaptive_mut` beats `full` significantly in seven of nine instances, with the two exceptions, both at 50x50, attributable to the reduced 10-seed subsample used there rather than to a vanished effect: the mean moves in the beneficial direction in all nine instances without exception. The effect is largest where Section 6.5's convergence curves show the plateau setting in early: -5.7% at 20x20, 40% obstacles ($p = 0.005$) and -4.6% at 50x50, 30% obstacles ($p = 0.043$).

Because `adaptive_mut` bundles two mechanisms, a mutation-rate increase and a population-reinjection step, trig-

TABLE IV

GA ABLATION: MEAN TURN COUNT AND WILCOXON P-VALUE (IN PARENTHESES, PAIRED BY SEED) AGAINST THE `full` CONFIGURATION. BOLD DENOTES $p < 0.05$; A DASH INDICATES FEWER THAN 5 NON-ZERO PAIRED DIFFERENCES. 50×50 ROWS USE A REDUCED 10-SEED SUBSAMPLE (SEEDS 0–9); SOLVABLE SEEDS WITHIN THAT SUBSAMPLE ARE 10, 9, AND 2 AT 10/30/50% OBSTACLE DENSITY RESPECTIVELY, SO THE DASHES AT 50% REFLECT SAMPLE SIZE RATHER THAN A NULL RESULT.

Instance	full	no_astar_init	no_elitism	adaptive_mut	adaptive_pm_only	reinject_only
10x10, 10% obs	1.73	2.33 (**0.044**)	1.77 (—)	1.57 (**0.043**)	1.70 (—)	1.53 (**0.028**)
10x10, 20% obs	2.83	3.03 (0.262)	2.93 (—)	2.66 (**0.043**)	2.79 (—)	2.66 (**0.043**)
10x10, 30% obs	4.25	3.96 (0.203)	4.50 (—)	3.61 (**0.003**)	4.25 (—)	3.71 (**0.008**)
20x20, 10% obs	3.07	3.43 (0.233)	3.07 (—)	2.83 (**0.018**)	2.97 (—)	2.73 (**0.008**)
20x20, 20% obs	5.70	5.77 (0.856)	5.73 (—)	5.27 (**0.003**)	5.63 (—)	5.27 (**0.003**)
20x20, 40% obs	10.48	10.84 (0.183)	10.72 (—)	9.88 (**0.005**)	10.40 (—)	9.80 (**0.003**)
50x50, 10% obs*	7.80	7.50 (0.447)	7.80 (—)	7.50 (—)	7.80 (—)	7.40 (—)
50x50, 30% obs*	21.78	21.44 (0.600)	22.44 (—)	20.78 (**0.043**)	21.56 (—)	21.33 (—)
50x50, 50% obs*	42.50	43.50 (—)	43.00 (—)	40.50 (—)	42.50 (—)	40.50 (—)

gered by the same stagnation signal, into one ablation arm, it cannot on its own say which piece is doing the work. Splitting it into `adaptive_pm_only` and `reinject_only` settles the question. `adaptive_pm_only`, the mutation-rate increase in isolation, produces fewer than five non-zero paired differences from `full` in all nine instances, the same untestable pattern `no_elitism` shows: boosting p_m alone changes almost nothing. `reinject_only`, the population-reinjection step in isolation, reaches significance in six of nine instances, reproducing nearly all of `adaptive_mut`'s effect, and in three instances (10x10 at 10%, 20x20 at 10%, and 20x20 at 40% obstacles) `reinject_only`'s mean turn count is slightly lower than the combined `adaptive_mut` arm's. Reinjecting a stagnation-triggered fraction of the population with fresh, re-perturbed individuals, not the accompanying mutation-rate increase, is the specific mechanism behind the largest lever this study tests. The mutation-rate-boost side of this pairing corresponds to triggered hypermutation and the reinjection side to a fixed-fraction random-immigrants scheme, both first proposed to help a GA track a moving optimum rather than escape stagnation on a fixed one [9]; the result here, that the immigrants-style reinjection component carries the effect and the hypermutation-style rate increase does not, is specific to this stagnation setting and should not be read as a general claim about the two mechanisms' relative value in the dynamic-optimization problems they were originally designed for.

Table 5 reports the equivalent ablation for ACO's `restart_dead_ends` configuration, described in Section 4.7, and is discussed together with the transfer question it was designed to test in Section 6.6.

E. Convergence Behavior

Figures 8–10 plot best fitness against iteration for one representative seed on each grid size. All three algorithms converge within a small fraction of their allotted budget: on the 10 × 10 instance shown, GA and PSO reach their final fitness within the first few generations of a 300-generation run, and ACO does the same within a handful of its 200 iterations. The same pattern holds at 20 × 20 and 50 × 50. This is a direct consequence of the shared A*-seeded initialization from Section 4.2: every population starts already

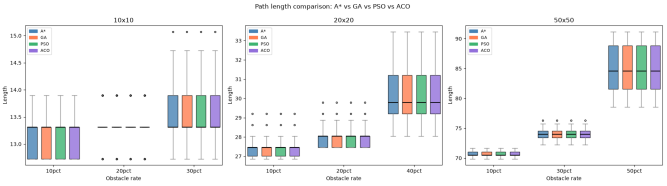

Fig. 6. Path length distributions for A*, GA, PSO, and ACO across all nine instances, grouped by grid size.

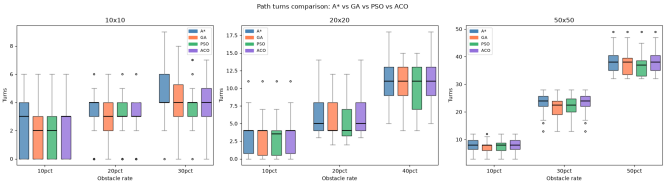

Fig. 7. Turn count distributions for A*, GA, PSO, and ACO across all nine instances, grouped by grid size.

length-optimal, so the remaining search only has to locate turn-reducing variants nearby, a much smaller search than finding a valid path from scratch, and it does not need hundreds of additional generations to do so.

The gap between algorithms is in where they plateau, not how fast they get there. On the 10 × 10 instance, GA and PSO settle to a best fitness near 5.93, while ACO settles near 6.13, a visibly higher (worse) plateau; the same ordering appears at 50 × 50, where GA settles near 30.42 against ACO's roughly 30.82. Because fitness is a weighted sum dominated by length (Equation 4) and length is fixed at the shared A* optimum for these instances, essentially all of the visible gap between plateaus is the $w_2 T(\pi)$ turn term, the same GA-over-ACO ordering Table 3 establishes in aggregate, now visible on individual optimization runs rather than only in the seed-averaged statistics. Section 6.4's ablation shows that this plateau is not fixed: reinjecting part of the population once stagnation is detected pushes GA's plateau lower still, exactly the pattern this convergence data would predict.

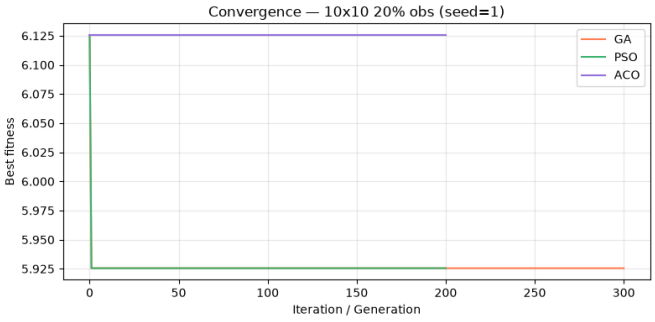

Fig. 8. Best-fitness convergence for GA, PSO, and ACO on a representative 10x10, 20%-obstacle instance.

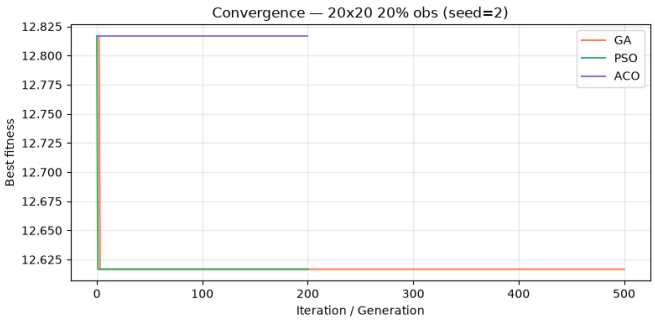

Fig. 9. Best-fitness convergence for GA, PSO, and ACO on a representative 20x20, 20%-obstacle instance.

F. Failure Modes

Four failure modes emerge from the raw data, none of them visible from the summary means alone.

Grid connectivity collapses well before universal disconnection. The number of seeds with a valid start-to-goal path, out of 30, falls from 30 at 10% obstacle density to 29 and 28 on the 10×10 grid, 30 and 25 on the 20×20 grid, and 30, 26, and 11 on the 50×50 grid as density rises through 10%, 30%, and 50%. The drop is not gradual: at 50% density on the largest grid, 19 of 30 randomly sampled instances, 63%, have no path at all, a sharp attrition consistent with random 8-connected grids sitting near a percolation-like connectivity threshold at that density, though establishing the precise threshold is outside the scope of this study. Every algorithm here inherits this ceiling identically, since all three raise the same error and abort whenever their A*-based initializer cannot find a seed path, so the reported $n = 11$ comparison at 50x50, 50% obstacles is not a weakness of any one method; it is the shared floor every method in this comparison stands on.

GA and PSO both pay for the same repair pathway, at different rates. On the 50×50, 10%-obstacle instance, mean runtime is 330.5 s for GA, 11.7 s for PSO, and 16.0 s for ACO, a roughly 20–28$\times$ gap between GA and the other two. Part of this is a deliberately unequal compute budget: at that grid size GA evaluates on the order of $120 \times 800 = 96{,}000$ offspring across its run, against PSO's $80 \times 200 = 16{,}000$ particle-updates and ACO's $50 \times 200 = 10{,}000$ ant-walks, a 6–9.6$\times$ difference in raw evaluation count that does not on its own account for a 20–28$\times$ runtime gap. The remainder is architectural: GA and PSO both route every offspring or particle through `simplify` and `repair`, and `repair` calls A* again internally for every broken segment a crossover or mutation produced, while ACO's ants can only ever step onto free, unvisited cells by construction and so never call `repair` at all, only the far cheaper `simplify`. The two algorithms that share the expensive repair pathway are the two slow ones; the algorithm that structurally avoids it is the fast one.

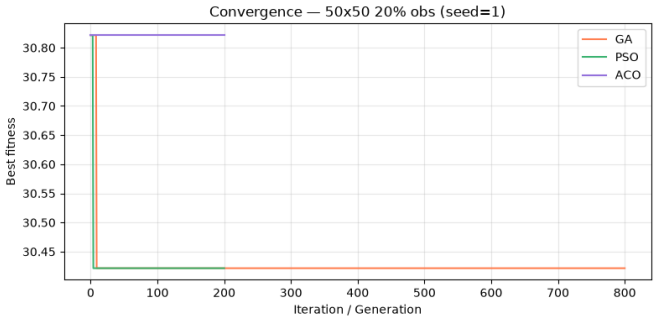

Fig. 10. Best-fitness convergence for GA, PSO, and ACO on a representative 50x50, 20%-obstacle instance.

PSO's runtime is non-monotonic in obstacle density, and the likely cause is that same repair pathway. PSO's mean runtime on the 50×50 grid is 11.7 s at 10% obstacles, 142.7 s at 30%, and 10.1 s at 50%, a more than tenfold spike at the middle density that neither its neighbor condition comes close to. A plausible mechanism, consistent with the repair-pathway explanation above but not directly confirmed by profiling data collected in this study, is that 30% obstacle density is dense enough to make many crossover-produced path splices land on an obstacle, triggering expensive local A* reroutes inside `repair`, while still being sparse enough that 26 of 30 seeds remain solvable at all; at 50% density, only the 11 most constrained, path-scarce instances survive the reachability filter, and those survivors may leave crossover and mutation little room to produce an invalid splice in the first place, since there are fewer alternative routes for a spliced path to wander into an obstacle along. This pattern does not appear in GA's runtime at the same three densities (330.5 s, 109.4 s, and 82.6 s, a monotonic decline), which is consistent with GA's much larger, size-scaled generation budget swamping whatever density-dependent repair cost PSO's fixed 200-iteration budget lets show through instead.

A GA-specific anti-stagnation fix transfers only where ACO does not need it. Section 4.7's `restart_dead_ends` configuration replaces a discarded, dead-ended ant with a perturbed-and-repaired copy of the colony's current best path rather than dropping it, the same reinjection idea Section 6.4 identifies as GA's strongest lever, transplanted onto ACO's per-ant construction step. Table 5 reports the result against the ACO baseline of Table 3, paired by seed.

TABLE V
ACO ABLATION: MEAN TURN COUNT AND WILCOXON P-VALUE VS. BASELINE, PAIRED BY SEED. A DASH INDICATES FEWER THAN 5 NON-ZERO PAIRED DIFFERENCES.

Instance	baseline	restart	p
10x10, 10% obs	1.83	1.63	0.076
10x10, 20% obs	2.90	2.76	0.237
10x10, 30% obs	4.14	4.04	0.484
20x20, 10% obs	3.23	3.23	—
20x20, 20% obs	6.07	6.07	—
20x20, 40% obs	10.96	10.92	—
50x50, 10% obs	7.97	7.97	—
50x50, 30% obs	23.00	23.00	—
50x50, 50% obs	38.73	38.73	—

At the two instances where ACO ties or trails A*, 20x20 at 40% obstacles and 50x50 at 50% obstacles, `restart_dead_ends` has no measurable effect: the mean turn count is unchanged at both, with zero non-zero paired differences at either instance. Where the fix does show a (non-significant) improvement is on the small, sparse 10x10 grids, the same instances where ACO was already beating A* significantly in Table 3.

The mechanism is legible directly from the implementation. `restart_dead_ends` only has something to work with when a dead-ended ant's replacement path can differ meaningfully from the ones the colony is already finding: on a sparse grid, several comparably short routes typically survive to the goal, so a perturbed-and-repaired restart can land on a different, sometimes better, path than the one that dead-ended. On the dense, large instances that collapse toward a single viable corridor, as the connectivity analysis above shows, there is essentially one route through the surviving free cells, so perturbing and repairing from the current best just reconstructs the same walk the colony had already found; a fix built around route diversity has nothing to diversify into. That is a mechanistic answer to the transfer question this study set out to ask, not just a null result: the same reinjection idea that is GA's strongest lever transfers exactly where ACO does not need it and does nothing where it does. Prior ACO-immigrants work gates effectiveness on environmental change rate [11]; here the gating variable is route diversity within a static grid.

G. Design Framework Versus Empirical Results

This project's design framework set out qualitative expectations for each algorithm before any experiment ran, and the results let two of those three be checked directly. GA was flagged as slow on large grids and dependent on repair operators; both hold exactly as predicted, with GA the most expensive algorithm at every grid size and the repair pathway the direct cause identified in Section 6.6. PSO was flagged as prone to local optima in dense obstacles and best suited to low-density maps; the turn-count results only partly support this, since PSO now reaches significance across all nine density conditions tested (Section 6.3), a broader competence than "best for low obstacle rate" alone would predict, even though its runtime (Section 6.6) does spike at the middle obstacle density on the largest grid.

ACO is where the design-stage expectation and the data disagree most sharply. The framework flagged ACO as memory-intensive and slow on small grids, best suited to large grids. The results run the other way on both counts: ACO is the cheapest or near-cheapest algorithm at every grid size, including a wide margin at 50 $\times$ 50 (16.0 s against GA's 330.5 s), and its one length-optimality failure (Section 6.2) and weakest turn-reduction performance (Section 6.3) both showed up on small grids or the two densest large-grid conditions, not on large grids generally. The likely explanation sits in the implementation rather than in grid size itself: a per-cell rather than per-edge pheromone field keeps memory and runtime low regardless of N, and the absence of a 2-opt correction step means ACO's search quality degrades wherever ants tend to dead-end, a property tied to obstacle density and search-space fragmentation rather than to cell count on its own.

One design-stage fix also never made it into the GA benchmarked in Table 3. The framework names premature convergence on large grids as a known risk and recommends an adaptive mutation rate or reinjecting the bottom 10% of the population with fresh individuals once diversity drops. The GA reported in Sections 6.2 and 6.3 does neither: p_m is a fixed 0.05 for the full run, with no diversity check and no reinjection step. The convergence curves in Section 6.5 show exactly the symptom this fix targets, all three algorithms flatten within a handful of iterations against a budget of up to 800, well ahead of the generation-200 mark the framework treats as a typical convergence point on a 50$\times$50 grid, and the benchmarked GA does so without the mitigation the framework specified for it. Section 6.4's ablation answers the question the original framework left open: reinjection, the specific mechanism the design-stage document called for, is the single strongest lever this study tests, beating the benchmarked GA significantly in seven of nine instances. That answer arrives with two qualifications the framework's original recommendation did not anticipate. First, the ablation isolates reinjection from the accompanying mutation-rate increase and finds the rate increase alone contributes almost nothing, so the framework's two-part recommendation is only half right. Second, per Section 6.6, transplanting the same reinjection idea onto ACO rather than GA does not push ACO to a lower plateau on the instances where it struggles; the mechanism is specific to how it interacts with GA's crossover and selection, not a general-purpose fix for any of the three algorithms benchmarked here.

VII. CONCLUSION

Sharing one representation, one fitness function, and one repair pipeline across GA, PSO, and ACO turns this comparison into a test of search strategy alone, and a follow-up ablation on that same shared codebase turns the comparison's headline result into a specific, attributable mechanism rather than an algorithm-level label. All three EAs match A*'s optimal path length on almost every instance, so length was never really in contention; turn count is where they separate, and GA's advantage there is not well explained by the elitism the original design attributed it to. The ablation shows that

removing top-5% elitism changes too few seeds per instance to test in any of the nine conditions, that warm-start A*-seeded initialization contributes a small and inconsistent amount, and that a stagnation-triggered mutation-rate boost combined with population reinjection, specified in the original design framework but never implemented in the benchmarked GA, is the single largest lever tested, reaching significance in seven of nine instances; splitting that combined mechanism further attributes the effect specifically to reinjection, not the accompanying mutation-rate increase. The same reinjection idea, transplanted onto ACO's ant-construction step, has no measurable effect on the two instances where ACO ties or trails A*, and a route-diversity argument explains why: `restart_dead_ends` only has an alternative route to fall back on when more than one viable path survives, and that condition does not hold on the dense, large grids where ACO needed the help. With the canonical A* baseline behind Table 3, GA cuts mean turns significantly against A* on eight of nine benchmark instances, missing only the smallest-sample condition (50x50 at 50% obstacles); PSO does so on all nine; and ACO on two, tying rather than losing to A* at the two dense, large-grid instances discussed throughout this paper.

Runtime tells a second, independent story that the ablation does not touch. GA and PSO both pay for the same A*-based repair step on every offspring or particle, which makes them the two expensive algorithms at large grid sizes, while ACO's construction process can never generate an invalid path in the first place and stays the cheapest of the three throughout. That ranking inverts a design-stage expectation worth restating plainly: this project's own framework predicted ACO would be the slow, memory-heavy option best reserved for large grids, and the data says the opposite on both counts, cheapest throughout and weakest exactly where grids get large and dense.

The comparison has real limits. Every grid here is static and fully known in advance; none of the three algorithms sees a moving obstacle or a partially observed map, so this study measures planning quality under conditions friendlier than most deployed robots face. The benchmark also stops at 30 seeds per instance, 10 at 50×50 for the ablation arms specifically, and nine grid/density combinations, a matrix chosen for breadth across grid size and obstacle density rather than depth at any single point on that grid, and the PSO runtime anomaly reported in Section 6.6 is a pattern in the data rather than a mechanism confirmed by direct profiling. A further limitation sits inside the comparison's main result rather than the ablation itself: the GA reported throughout Sections 6.2 and 6.3 is still the fixed-mutation-rate, no-reinjection design; `reinject_only` was evaluated as a controlled comparison against that design, not adopted as the primary algorithm behind Table 3. Table 3's GA numbers therefore understate what a GA built around this study's own strongest ablation finding could achieve, an adoption question deliberately left for future work rather than folded into the main comparison after the fact.

Two directions from the related work in Section 2 point past these limits, and a third follows directly from the ablation itself. Qin et al. [6] describe hierarchical designs that hand a static global plan, of exactly the kind this study benchmarks, to an online learning policy for dynamic obstacle avoidance, which would let a GA- or PSO-planned baseline stay useful once obstacles start moving rather than requiring a full replan. Puente-Castro et al. [7] extend a single-agent cell-grid GA to UAV swarms with per-vehicle scheduling, a natural next step for the shared representation used here, since every path in this study is already a plain sequence of grid cells that a multi-agent scheduler could coordinate across several such sequences at once. The third direction is to adopt reinjection as GA's default configuration rather than an ablation arm, and to test whether a differently triggered or differently sized reinjection step, rather than route diversity itself, is the missing ingredient for a GA-style anti-stagnation fix to help ACO on the dense, large grids where it currently does not.

References

[1] J. Zhu and D. Pan, "Improved Genetic Algorithm for Solving Robot Path Planning Based on Grid Maps," *Mathematics*, vol. 12, no. 24, art. 4017, 2024.

[2] L. Zheng, W. Yu, G. Li, G. Qin, and Y. Luo, "Particle Swarm Algorithm Path-Planning Method for Mobile Robots Based on Artificial Potential Fields," *Sensors*, vol. 23, no. 13, art. 6082, 2023.

[3] J. Ma, Y. Liu, S. Zang, and L. Wang, "Robot Path Planning Based on Genetic Algorithm Fused with Continuous Bezier Optimization," *Computational Intelligence and Neuroscience*, vol. 2020, art. 9813040, 2020.

[4] H. T. Najm, N. S. Ahmad, and A. S. Al-Araji, "Enhanced Path Planning Algorithm via Hybrid WOA-PSO for Differential Wheeled Mobile Robots," *Systems Science & Control Engineering*, vol. 12, no. 1, art. 2334301, 2024.

[5] Y. Tang, M. A. Zakaria, and M. Younas, "Path Planning Trends for Autonomous Mobile Robot Navigation: A Review," *Sensors*, vol. 25, no. 4, art. 1206, 2025.

[6] H. Qin, S. Shao, T. Wang, X. Yu, Y. Jiang, and Z. Cao, "Review of Autonomous Path Planning Algorithms for Mobile Robots," *Drones*, vol. 7, no. 3, art. 211, 2023.

[7] A. Puente-Castro, E. Fernandez-Blanco, and D. Rivero, "Genetic Algorithm Based System for Path Planning with Unmanned Aerial Vehicles Swarms in Cell-Grid Environments," arXiv preprint arXiv:2412.03433, 2024.

[8] J. N. Hooker, "Testing Heuristics: We Have It All Wrong," *Journal of Heuristics*, vol. 1, no. 1, pp. 33–42, 1995, doi: 10.1007/BF02430364.

[9] H. G. Cobb and J. J. Grefenstette, "Genetic Algorithms for Tracking Changing Environments," in *Proc. 5th International Conference on Genetic Algorithms*, Morgan Kaufmann, San Francisco, CA, 1993, pp. 523–530.

[10] T. Stützle and H. H. Hoos, "MAX–MIN Ant System," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889–914, 2000, doi: 10.1016/S0167-739X(00)00043-1.

[11] M. Mavrovouniotis and S. Yang, "Ant Colony Optimization with Immigrants Schemes in Dynamic Environments," in *Parallel Problem Solving from Nature – PPSN XI*, LNCS vol. 6239, Springer, Berlin, Heidelberg, 2010, pp. 371–380, doi: 10.1007/978-3-642-15871-1_38.

[12] J. Derrac, S. García, D. Molina, and F. Herrera, "A Practical Tutorial on the Use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011, doi: 10.1016/j.swevo.2011.02.002.

[13] G. Rudolph, "Convergence Analysis of Canonical Genetic Algorithms," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 96–101, 1994, doi: 10.1109/72.265964.

Junior Charlie is currently an undergraduate student majoring in Artificial Intelligence at Nanjing University of Information Science and Technology. His research interests include AI-based anomaly detection and digital twin systems for critical infrastructure, with a particular focus on translating real-time sensor data into autonomous monitoring and control decisions for smart city applications.