# Genetic Algorithm Tuning of Policy Parameters in a Cooperative Multi-Agent Environment

Ron Dahlgren
ron@sw.gy

May 15, 2026

**Abstract**

I report on a multi-configuration study of genetic algorithm (GA) tuning applied to my scripted policy in the CogsGuard cooperative multi-agent environment. Across seven GA runs spanning two policy classes and six parameter schemas (19 to 65 curated knobs), I evaluated approximately 60,000 candidate policy configurations using a tournament-style fitness function with episode-level rewards. In five of seven runs comprising the primary analysis cohort, GA produced statistically significant improvements in mean policy fitness; one run showed no improvement, providing an empirical counterpoint that I examine in detail. Beyond the headline question of whether GA improves fitness, I characterize how team composition (the share of cogs controlled by the tuned policy, and the identity of teammate policies) shapes both the absolute reward attained and the marginal benefit of additional control. I also extract concrete case-study findings about which parameter dimensions the GA most heavily exploited.

## 1 Introduction

CogsGuard is a cooperative multi-agent environment in which a team of eight *cogs* attempts to control a dangerous environment under partial observability. Multiple scripted policy implementations exist; new implementations are submitted by independent authors and ranked on a shared leaderboard. My own policy exposes a set of tunable parameters; in this study, ranging from 19 to 65 numeric or categorical *knobs* per policy version, whose ideal settings are not obvious a priori. Other authors' policies are opaque to me. I investigate whether a straightforward genetic algorithm is an effective tool for finding good knob settings without policy-specific tuning machinery.

A complication particular to cooperative settings is that a policy's fitness depends on the team it plays with. A policy that excels when controlling a minority of cogs may behave differently when controlling a majority, and performance may further depend on which other policies make up the rest of the team. My analysis therefore needs to treat *share* (the fraction of cogs controlled by the tuned policy, evaluated at 2/6, 4/4, and 6/2) and *teammate profile* (the policy class of the other cogs) as first-class dimensions.

The contributions of this paper are:

1. **Headline.** Evidence that GA reliably improves policy fitness across diverse parameter schemas (Section 3). In four of five long-running configurations, the mean fitness of the population improved by between +0.73 and +6.83 reward units, with 95% bootstrap CIs excluding zero. One configuration failed to improve.

2. **Team-composition effects** (Section 4). In the successful runs, increasing the tuned policy's share from 2/6 to 6/2 raised per-cog reward by +2.4 to +6.4, and this benefit was much larger with weak teammates than with strong ones, a clear share-by-profile interaction.
3. **Discovered tuning strategies** (Section 5). I extract specific parameter dimensions on which the GA strongly converged, in each case explainable in role/strategy terms without reference to specific policy implementations.

# 2 Methods

## 2.1 Evaluation pipeline

Each candidate parameter setting (a *genome*) is evaluated by running the policy in CogsGuard for a fixed number of steps and recording the mean per-cog reward over five episodes. Within a GA generation, candidates pass through a sequence of evaluation *rounds*: an initial round at a short step budget, followed by progressively longer rounds with culling of the worst-performing genomes between rounds (so that surviving genomes get more evaluation budget). All runs in this study use either a two-round pipeline ($r_0 \to r_1$, culling half between them) or a three-round pipeline ($r_0 \to r_1 \to r_2$). Round-step values vary by run; see Table 1.

Each genome is evaluated under three share settings (2/6, 4/4, 6/2) and multiple teammate-profile settings. A small *hall-of-fame* (HoF) mechanism re-introduces the previous generation's best genome as a teammate-profile choice in later rounds, providing an internal benchmark that adapts as the GA progresses.

The GA's selection signal is one of two *proxy* statistics computed over the final round's episode-level rewards: the population mean (`mean`) or the 25th percentile (`p25`). Both statistics are recorded for every generation regardless of which one the GA optimized, so all trajectory plots in this paper show both metrics.

## 2.2 Cohort selection

Twelve GA runs were performed over the study period. I retained seven runs as the analysis cohort:

- **Tier A** (five runs): each completed seven or more generations, sufficient for trajectory analysis.
- **Tier B** (two runs): three to four generations completed, retained for share/profile analyses but excluded from cross-run trajectory aggregation.
- **Excluded** (five runs): runs that terminated after fewer than three generations and one duplicate (a partial-write predecessor of a "salvaged" version of the same GA process).

Table 1 gives the cohort configuration.

## 2.3 Statistical methods

All confidence intervals reported below are percentile bootstrap intervals at 95%, computed from $R = 3000$–$5000$ resamples. For estimates of mean fitness within a generation I resample the population members at that generation directly; for estimates of mean reward within a $(\text{run}, \text{share}, \text{profile})$ cell

**Table 1:** Tier-A cohort configurations. Population sizes and generation budgets varied across experiments. The `proxy` column lists the fitness statistic the GA used for selection.

| Run | Pop. | Gens (cfg/reach) | Proxy | Knobs | Round steps | Mission |
|---|---|---|---|---|---|---|
| `salvaged` | 32 | 20 / 15 | p25 | 19 | 3000, 4000, 7000 | `machina_1` |
| `v3` | 32 | 12 / 7 | mean | 28 | 5000, 8000 | `machina_1.clips` |
| `v3_nav` | 48 | 12 / 12 | mean | 36 | 4000, 6000 | `machina_1.clips` |
| `v5` | 128 | 10 / 10 | mean | 50 | 6000, 8000 | `machina_1.clips` |
| `v8` | 128 | 15 / 11 | mean | 65 | 6000, 8000 | `machina_1.clips` |

I resample the trial-level rewards within that cell. Cross-run summary statistics (e.g. median of run-level improvements) bootstrap over the five Tier-A runs as the unit. Knob-by-knob importance is reported as the Fisher-$z$-averaged within-generation Spearman correlation between knob value and fitness, with the within-generation restriction isolating the signal that high-fitness genomes have different knob values from low-fitness genomes *at the same generation*, independent of the overall direction in which the GA pushed the knob.

I do not pool absolute fitness across runs (different policy classes, different missions, different round-step budgets make absolute comparisons unsafe). All cross-run claims are about *relative improvement* or about share/profile contrasts measured within each run on its own scale.

## 3 Headline: GA improves fitness across configurations

Figure 1 shows the per-generation fitness trajectory for each of the five Tier-A runs. Each panel plots three statistics: population best, mean, and 25th percentile, with the GA's selection proxy emphasized in bold weight. A shaded ribbon shows the 10th–90th percentile of the per-generation population fitness distribution.

Four of the five Tier-A runs show clear monotonic improvement in all three metrics. The `salvaged` run shows a sharp two-generation gain followed by plateau-with-noise, characteristic of a GA that has found a local optimum and is exploring around it. The `v3` run, examined in detail below and in Appendix A, does *not* improve: mean fitness drifts slightly downward over the six generations evaluated. The `v3_nav` run improves modestly but persistently. The `v5` run improves smoothly over its ten generations. The `v8` configuration is the strongest case: mean fitness rose from 15.5 at generation 0 to 22.3 by generation 9, with the distribution narrowing as it climbed.

Figure 2 summarizes the per-run improvement in mean fitness from generation 0 to the final non-stunted generation, with 95% bootstrap CIs. The cross-run median improvement is +3.49 reward units (95% CI: $[-0.60, +6.83]$). The wide CI reflects the small number of runs ($n = 5$); the per-run CIs are narrower because each is computed from many population members. Four of five runs have CIs that exclude zero on the positive side; `v3` has a CI that excludes zero on the negative side.

**The `v3` failure.** `v3`'s mean fitness drops from 5.78 to 5.18 over six generations, with a 95% CI for the change of $[-0.85, -0.36]$ that excludes zero. This is genuine negative progress, not a noise event. The configuration that *immediately followed* `v3`, named `v3_nav`, used the same starting genome but added eight additional navigation knobs (28 → 36 curated knobs) and a larger population (32 → 48), and it reached modest positive improvement (+0.73 in mean fitness; 95% CI $[+0.52, +0.94]$). The

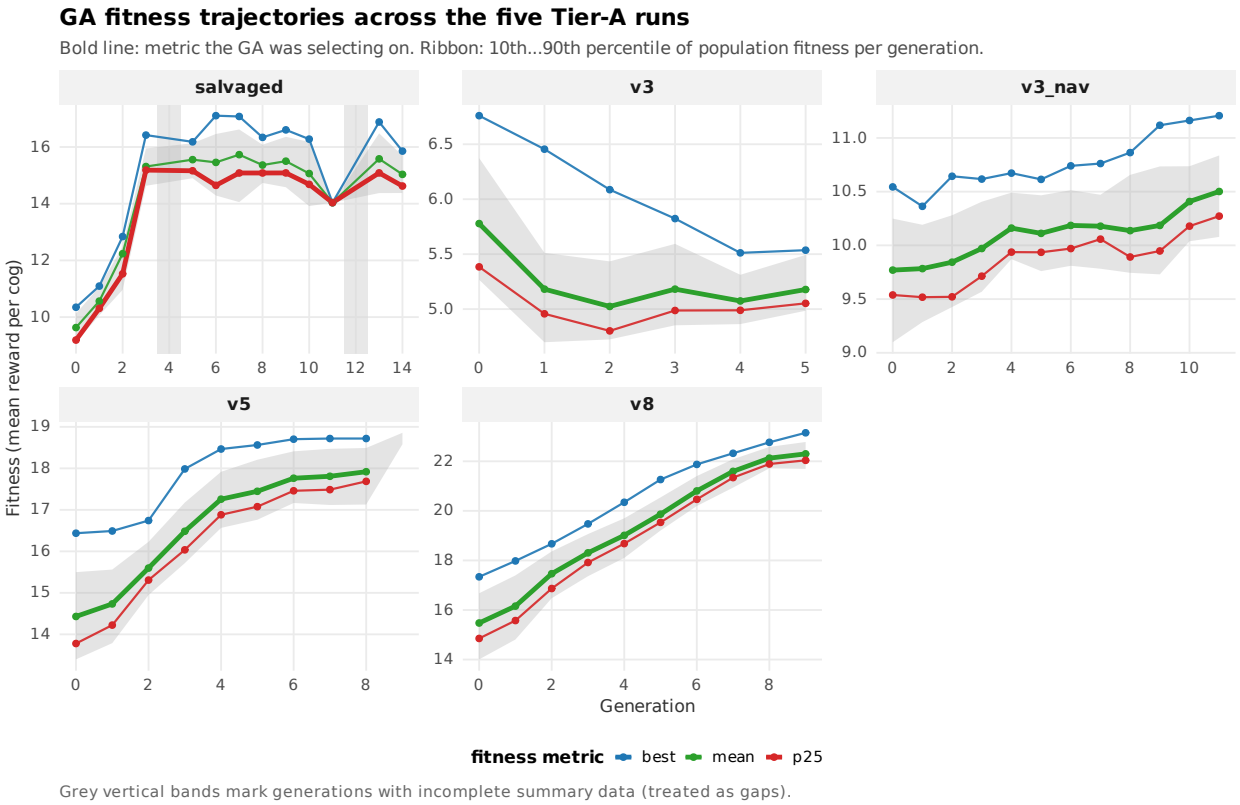



**Figure 1:** Per-run fitness trajectories. Bold line: the metric the GA was selecting on. Grey vertical bands in the `salvaged` run mark two generations where summary statistics were incomplete due to interrupted writes during a long resume process; those generations are excluded from trajectory aggregation but their trial-level data is retained for share/profile analyses.

simplest reading is that `v3`'s search space did not contain a productive direction the GA could climb, while `v3_nav`'s enlarged search space did. Appendix A provides the diversity and fitness-distribution comparison.

**Takeaway.** The cross-run evidence is consistent with the claim that GA tuning works for this class of policies: when the search space contains usable directions, the GA reliably finds and exploits them. When the search space does not, the GA does not manufacture progress. This is a failure mode that should be checked for in any new policy by running with multiple seeds before committing significant compute.

# 4 Team composition shapes what the GA can extract

## 4.1 Share effect: more cogs, more per-cog reward

Figure 3(a) shows the mean per-cog reward at each of the three share settings, computed from final-round trials in the last three generations of each Tier-A run (the converged-genome regime) and excluding HoF profiles. In four of five runs, per-cog reward *increases* with the share controlled by the tuned policy, by between +2.35 and +6.43 reward units when moving from 2/6 to 6/2. The

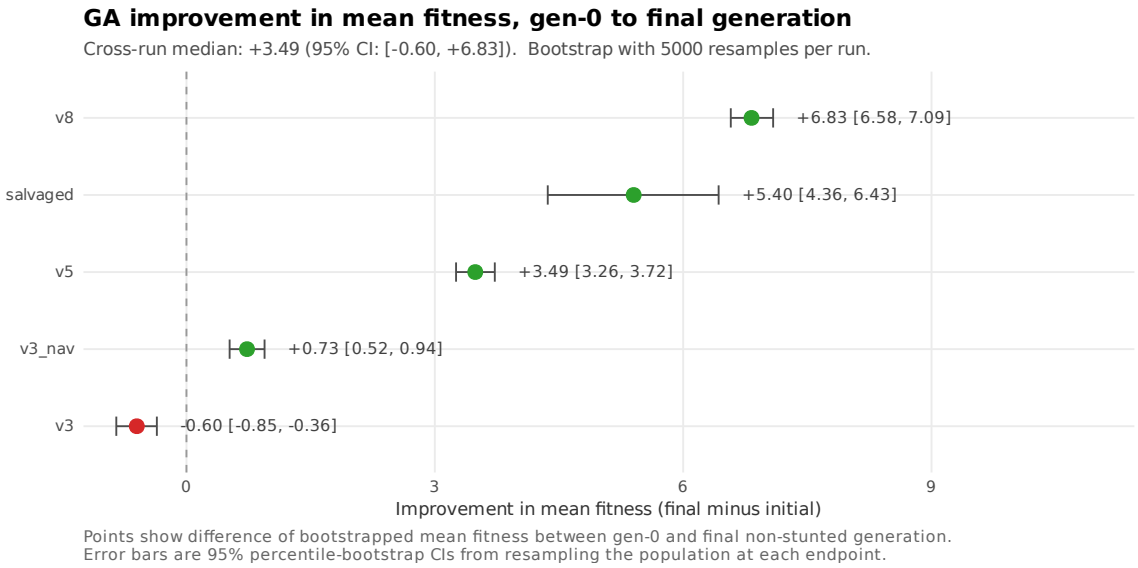
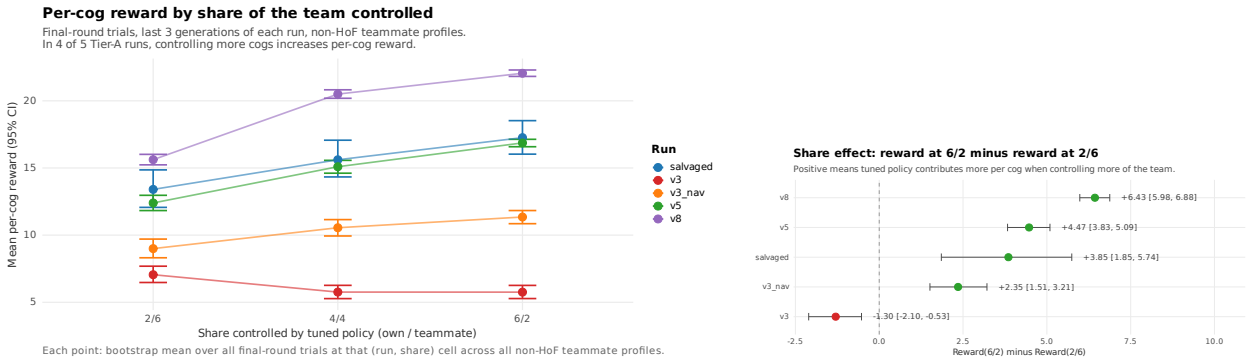



**Figure 2:** Improvement in mean fitness from gen-0 to final generation, with 95% bootstrap CIs. The cross-run median (computed across run-level deltas with bootstrap resampling over runs) provides the headline summary statistic.

exception is `v3` (Figure 3(b): $-1.30$, 95% CI $[-2.10, -0.53]$), which is consistent with its failure to converge on a good policy: a poorly-tuned policy in greater numbers makes things worse, not better.




**(a)** Mean per-cog reward at each share.

**(b)** Reward at 6/2 minus reward at 2/6, per run.

**Figure 3:** Share effect on per-cog reward. The metric is *per cog*, not total team reward, so positive slopes indicate the tuned policies cooperate well with each other: a 6/2 team of tuned policies is more than three times as effective per cog as a 2/6 team.

That the metric is reward *per cog* matters: if controlling more cogs simply meant more total reward, the share effect would be expected to be zero on a per-cog basis. A positive per-cog share effect means the tuned policies are cooperatively effective with copies of themselves; they build on one another, not just stack independently.

## 4.2 Teammate-profile effect: who you are paired with matters

Figure 4 shows per-cog reward by teammate profile, faceted by share. Four teammate profiles are present in most runs: `cvc`, `mas`, `starter`, and `tom` (the `salvaged` and `v3_nav` runs lack `tom`). Across all 15 `(run, share)` cells in the Tier-A cohort, the `mas` teammate profile ranked first in every cell. The `starter` teammate profile ranked last in 7 of 15 cells. The ordering `starter` $\leq$ `cvc` $\approx$ `tom` $\leq$ `mas` is highly stable.

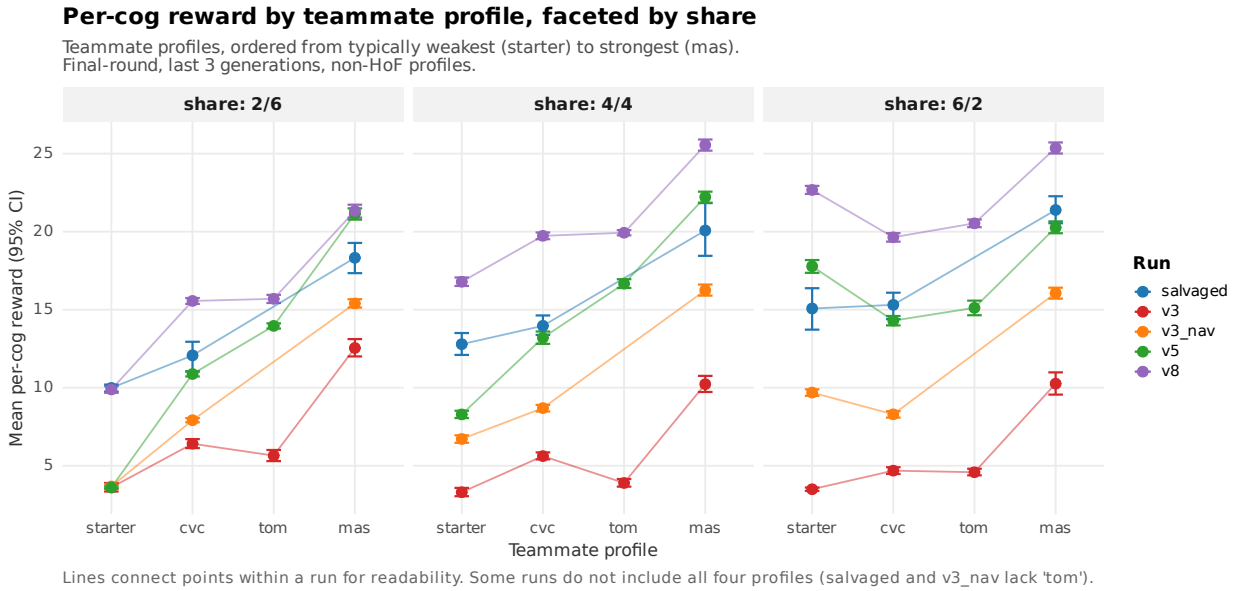



**Figure 4:** Per-cog reward by teammate profile, faceted by share. The `mas` profile is the strongest teammate by a wide margin at low share, the margin narrows at balanced share, and at high share (6/2) the choice of teammate matters less.

## 4.3 Share-by-profile interaction

The two effects above are not independent. Figure 5 plots the *share effect* (reward at 6/2 minus reward at 2/6) for each teammate profile separately. Across the four successful Tier-A runs:

- With `starter` teammates: mean share effect +9.5 reward units. Weak teammates drag the tuned policy down at low share; having more tuned cogs rescues the team.
- With `cvc` or `tom` teammates: mean share effect +2.8 to +3.0 reward units.
- With `mas` teammates: mean share effect +1.7 reward units. Strong teammates are already strong; the marginal benefit of swapping a strong teammate for a tuned cog is small.

**Takeaway.** The cooperative structure of CogsGuard means that "how well does the tuned policy perform" is not a single number. The deployment context matters: the same tuned genome can be most valuable in a minority-with-weak-teammates situation (large per-cog gain from each additional tuned cog) and least valuable in a minority-with-strong-teammates situation (the team is already near its ceiling). When evaluating policy submissions on a leaderboard, the chosen mix of teammates can significantly re-rank candidates.

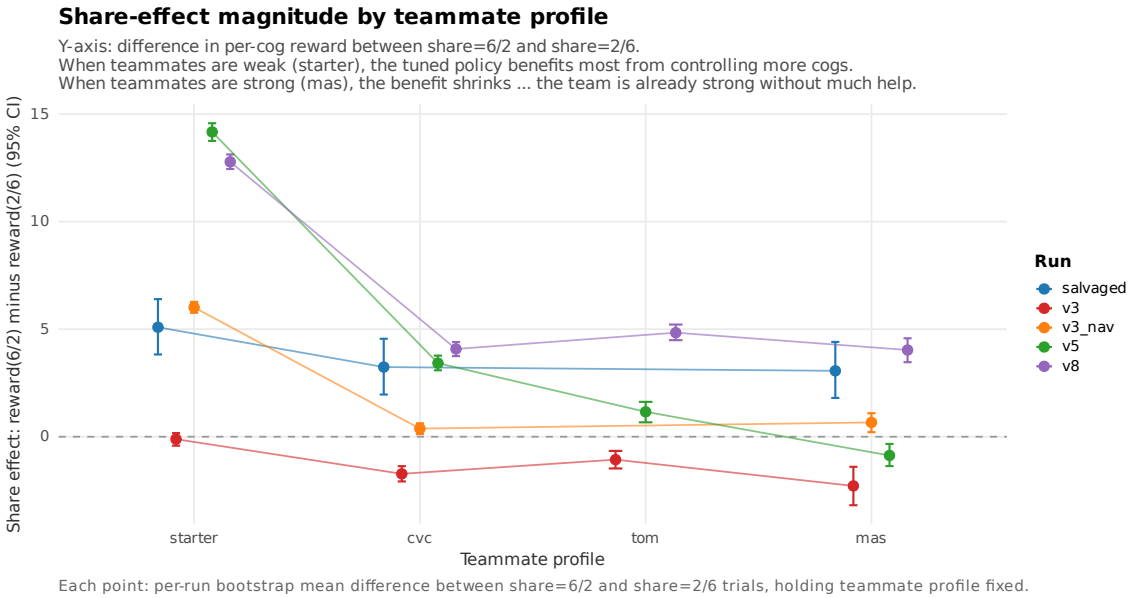



**Figure 5:** Share effect (reward at 6/2 minus reward at 2/6) for each teammate profile, per run. The fan-out at the `starter` end versus the convergence near zero at the `mas` end is the share-by-profile interaction.

# 5 Case studies: what did the GA discover?

I use the within-generation Spearman correlation analysis described in Section 2 to identify knobs the GA most strongly exploited. A knob with a large within-generation correlation tells me that at any given generation, genomes with that knob's value in one direction out-perform genomes with the value in the other direction, separate from the question of whether the GA happened to push the knob in that direction overall. Three case studies follow; each is drawn from a single run and described in role/strategy terms without policy-implementation specifics.

## 5.1 Navigation tabu length (run `v3_nav`)

`v3_nav`'s most consistent within-generation positive correlation is on the navigation tabu length parameter ($r = +0.19$, 12 generations). The median value increased from 9.5 at generation 0 to 19.0 at generation 11. The GA approximately doubled the length of the "recently-visited cells to avoid" window during pathfinding.

**Interpretation.** A longer tabu length forces the navigator to explore more aggressively, at the cost of sometimes taking longer paths. The GA's preference for longer tabu trails suggests that, in this environment, the cost of revisiting recently-traversed cells (and re-encountering the situations that drove the original visit) outweighs the path-length cost of forced detours.

## 5.2 Hold-phase team composition (run `v8`)

My policy divides a 10,000-step episode into three broad phases characterized as *expand* (early-game exploration and resource establishment), *hold* (mid-game steady-state operation), and *endgame* (late-game push for terminal reward). Several knobs control how cogs are allocated to specific roles

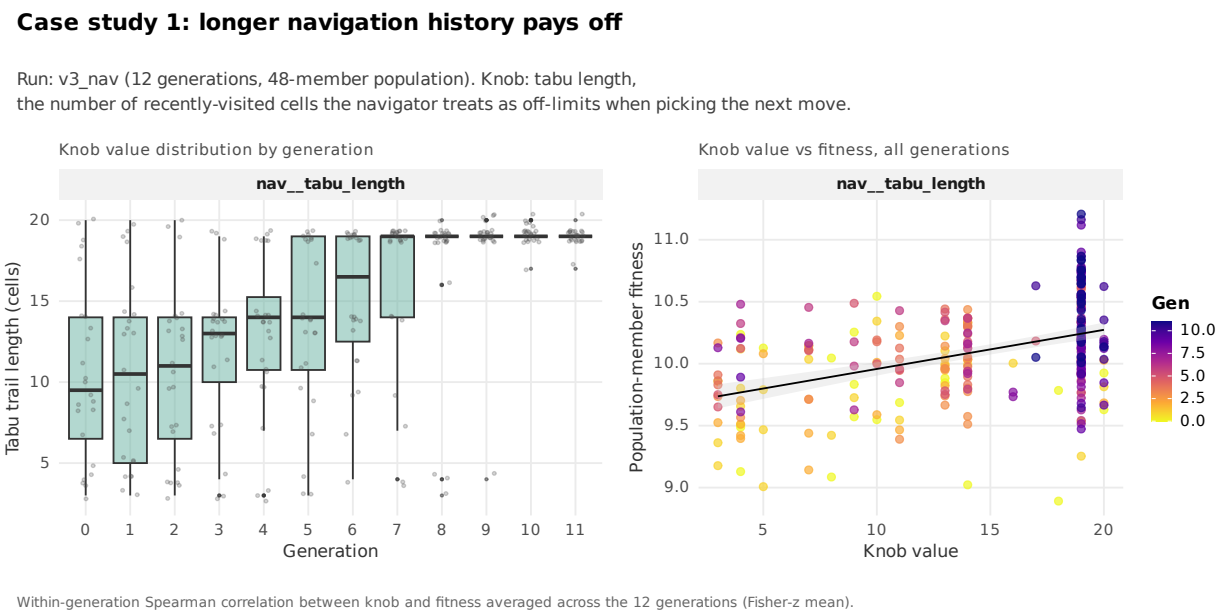



**Figure 6:** Case study: navigation tabu length. Left: per-generation distribution of the knob value; the GA pushed the value upward over time. Right: scatter of knob value versus per-genome fitness across all generations.

within each phase. Two roles are relevant here: the *aligner*, which converts neutral junctions into team-aligned ones, and the *scrambler*, which disrupts enemy-aligned junctions.

The strongest two within-generation correlations in `v8` are on these two roles during the hold phase. The percentage of cogs allocated to the scrambler role during the hold phase shows $r = -0.27$, while the percentage allocated to the aligner role in the same phase shows $r = +0.13$. The GA simultaneously decreased the scrambler allocation (early mean 33% $\rightarrow$ late mean 16% over 10 generations) and increased the aligner allocation (early 51% $\rightarrow$ late 73%).

**Interpretation.** The initial population was biased toward scrambler-heavy hold-phase compositions, and the GA discovered that this was suboptimal: more cogs in aligner roles outperformed scrambler-heavy compositions during the hold phase. This is intuitive in hindsight. Scrambling is most valuable when the environment is dynamic, while the hold phase is by definition the period when the team is consolidating its position. The GA found this gradient without being told to look for it.

## 5.3 Health-point risk tolerance (run `v8`)

A third case study from `v8` addresses the `hp_buffer_many_hearts` knob, which sets the minimum health-point reserve at which the policy is willing to engage in costly actions when its inventory of *hearts* is already replenished. Hearts are the currency used to scramble an enemy-aligned junction or align a neutral one; each heart costs seven of each resource type to produce, so they are an expensive and finite asset. The within-generation correlation with fitness is $r = -0.19$ (lower buffer correlates with higher fitness), and the GA cut the population mean from 15.6 at generation 0 to

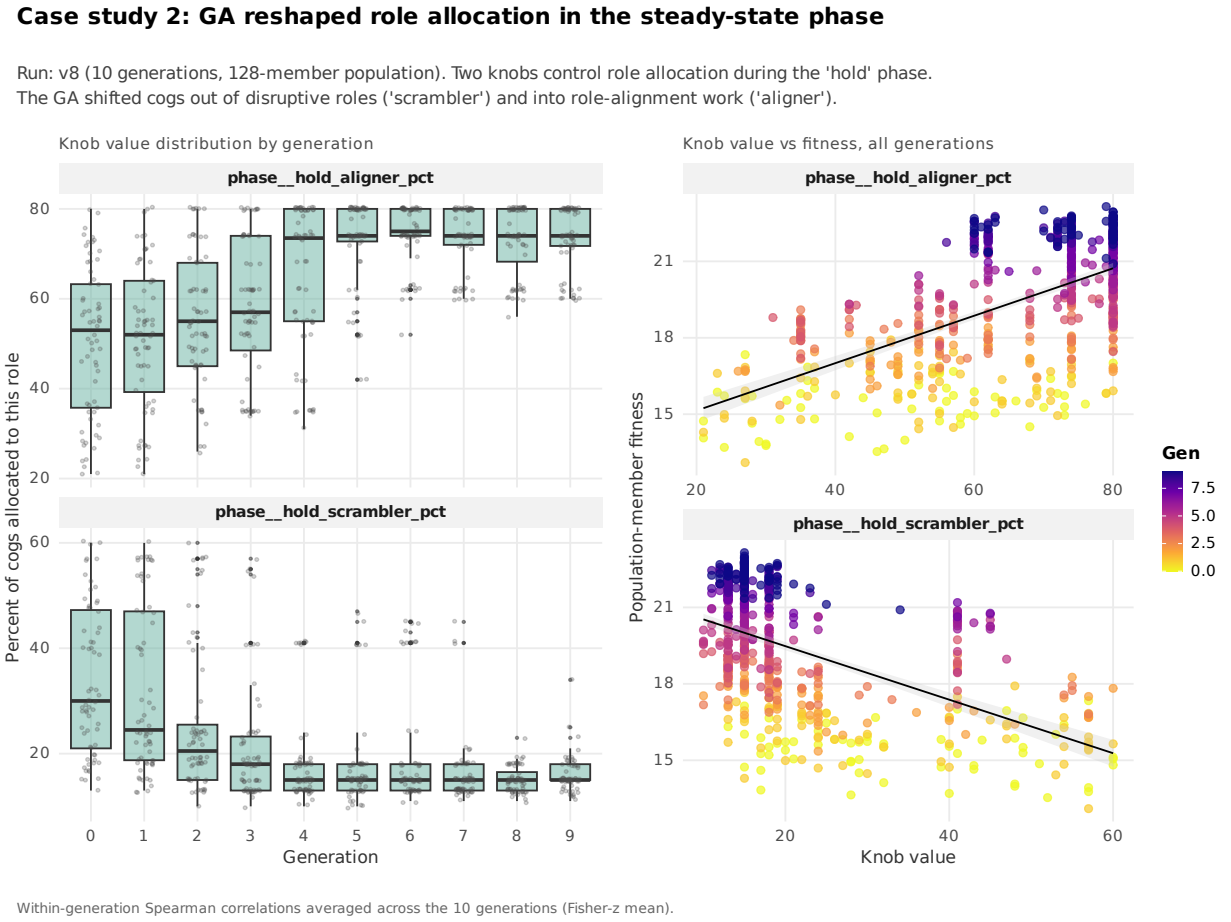



**Figure 7:** Case study: GA shifted cog role allocation during the hold phase. Each point is a genome in the final-round population; color denotes generation.

6.5 at generation 9, a 58% reduction.

**Interpretation.** The original tuning of the policy treated "many hearts" as a state in which to play safe (preserve HP for later). The GA found that this was conservative beyond what the fitness landscape rewarded: when the team is already heart-rich, holding additional hearts adds little value compared to spending them on junction alignment or scrambling actions that translate directly to reward. The discovery is small in isolation but representative of a class of findings the GA produces: identifying "defaults" in the policy that were never deliberately tuned and revealing the appropriate operating point.

**Why three case studies, not fifteen.** Many other knobs have non-trivial within-generation correlations (the per-run top-15 lists are available in supplementary material). The three studies above were selected not for being the strongest correlations but for being the most *interpretable in cooperative-environment terms* without reference to the specific policy implementation. The hold-phase composition is a team-strategy finding; the tabu length is a navigation-strategy finding;

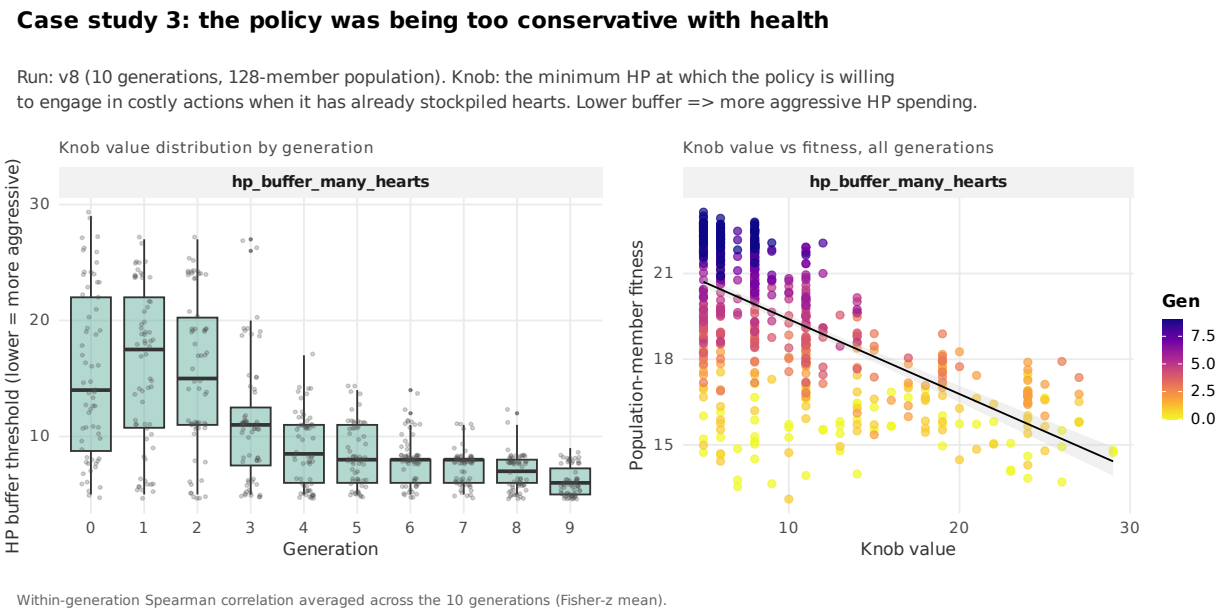



**Figure 8:** Case study: HP risk tolerance when carrying many hearts. The GA discovered that the policy was being too conservative with health when its heart inventory was already replenished, and pushed the conservation threshold sharply downward.

the HP buffer is a resource-spending finding. All three generalize as hypotheses that other policy authors could test on their own implementations.

# 6 Discussion

**When GA works, what it costs.** On the seven runs in my cohort, total fitness-evaluation wall-time ranged from ∼360 to ∼2340 hours of single-core CPU equivalent. Each successful run identified a better-tuned policy than its starting population, but the marginal generations toward the end of each run showed diminishing returns (Figure 1). The practical implication is that a budget of 5–10 generations on a population of 50–100 is sufficient to capture most of the available GA improvement, and there is little need to run for 20 generations in pursuit of small additional gains.

**Selection proxy: `mean` vs. `p25`.** Appendix B examines whether the choice of selection proxy affected the population's evolution. In the runs where the GA selected on mean, the P25–mean spread *also* narrowed across generations (a side-effect: as the mean climbs, the lower tail tends to climb with it). In the only run that selected on P25 (`salvaged`), the spread is noisier and wider on average, consistent with selecting on the lower tail rather than the center. For tournament-relevant ranking (where mean is the metric), selecting on mean is unsurprisingly more direct; whether the robustness advantage from P25 selection is worth the slower mean climb is implementation-dependent.

**Comparability across policies.** The runs in my cohort spanned two policy classes (`policy_mas.Mas` for `salvaged`; `policy_anticlips.AntiClipsPolicy` for the rest) and six knob schemas (the v3, v3_nav, v4, v5, v8, and earlier versions). That GA improved fitness on five of the seven runs, across these different configurations, supports the methodological claim, even though cross-run absolute-fitness comparisons would not be sound. The `v3` failure on one schema (and the immediate success of `v3_nav` on a closely related schema) provides a useful boundary on the methodology.

**Limitations.**

- Five Tier-A runs is a small sample for cross-run inference; the ± wide CI on the cross-run median improvement reflects this.
- Two of the seven runs (Tier B) reached only 3–4 generations before being terminated; their share/profile analyses contributed to the cross-cohort patterns but their fitness trajectories are not included in the headline figures.
- The teammate-profile axis included a Hall-of-Fame mechanism that introduces non-stationarity across generations; I excluded HoF profiles from the share/profile analyses but did not analyze the HoF-profile interaction in this paper.

**Conclusion.** GA is an effective and economical tool for tuning the parameters of scripted cooperative-environment policies. The methodology generalizes across different policy implementations and parameter schemas, with one observed failure case that was diagnosable and recoverable in the immediately-following experiment. Beyond the fitness improvements themselves, the GA's selection pressure provides interpretable signal about which parameter dimensions matter, surfacing strategy-level findings (role allocation, navigation behavior) that are useful independent of the optimization objective.

# A The `v3` failure in detail

Figure 9 compares the fitness distribution per generation for `v3` (the failed run) against `v3_nav` (the immediately-following run that used a similar configuration plus eight additional navigation knobs).

`v3`'s generation-0 population had wider fitness variance than `v3_nav`'s, with several outliers above 6.0. The GA's selection narrowed the population around a value close to the gen-0 median (5.7), losing the high-fitness outliers; it did not subsequently rediscover anything better. This is a recognizable pattern of *premature convergence*: when the search space lacks a productive direction the GA can follow, selection pressure can destroy initial diversity faster than mutation regenerates useful diversity, leaving the population locked around an unremarkable median.

# B Selection proxy and population spread

# C Within-trial and between-trial variance

For the high-fitness runs (`v5`, `v8`), within-trial standard deviation exceeds between-trial standard deviation, suggesting that for those runs the dominant source of noise is episode-to-episode variation

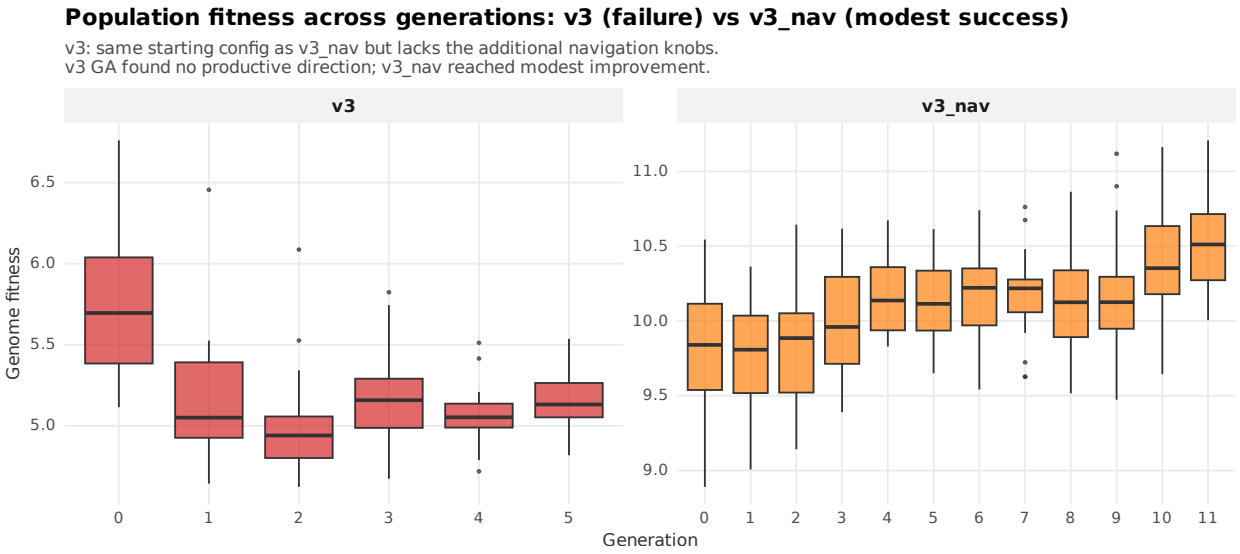



**Figure 9:** Population fitness across generations. The `v3` GA ran for six generations and produced no upward trend; `v3_nav` ran for twelve generations and produced a modest, persistent climb.

rather than between-trial random seeds. A higher episode count per trial (10–20) would tighten the per-trial estimates in those runs.

# D Per-round trajectory

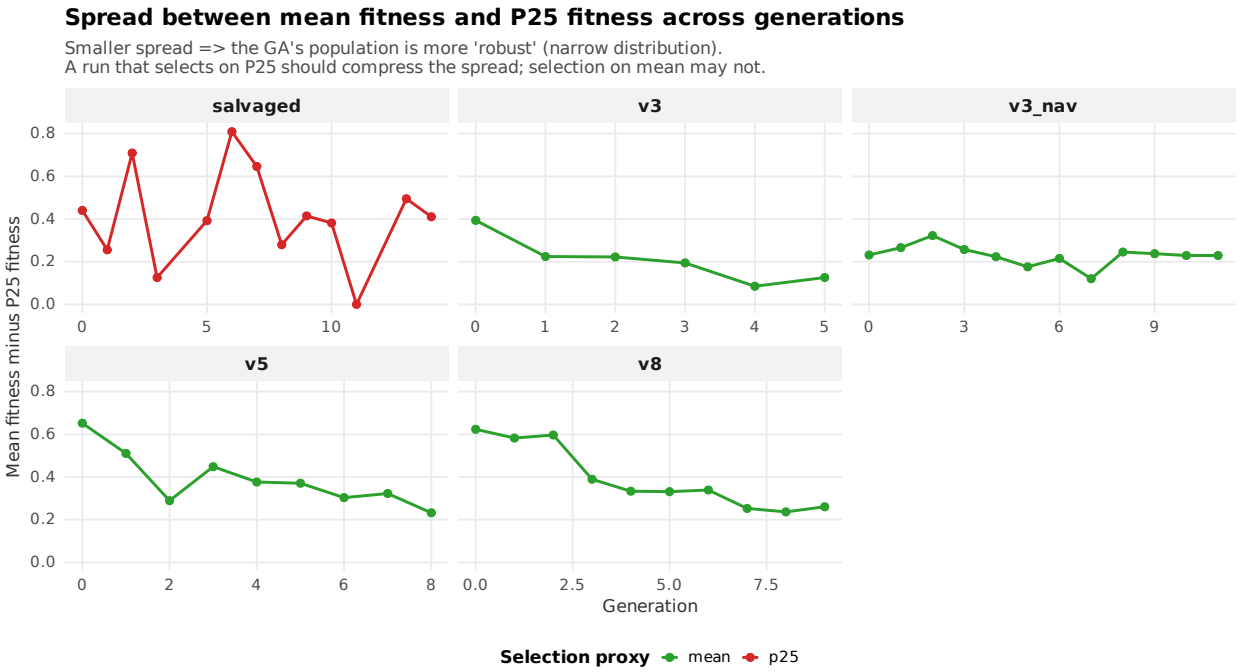



**Figure 10:** Spread between mean fitness and 25th-percentile fitness across generations, per Tier-A run. Color indicates which proxy the GA selected on. Mean-selected runs (`v3`, `v3_nav`, `v5`, `v8`) show the spread narrowing as the population mean climbs; the only P25-selected run (`salvaged`) shows a noisier, larger spread.

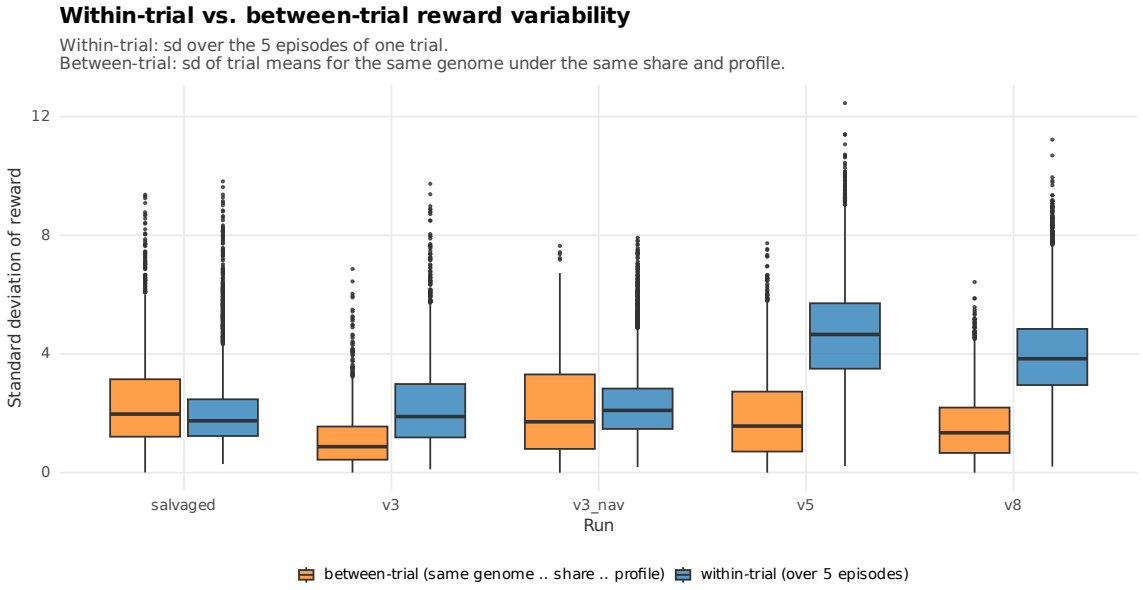



**Figure 11:** Within-trial reward standard deviation (over the five episodes of one trial) versus between-trial standard deviation (sd of trial means for the same genome under the same share and teammate profile). Where the within-trial sd substantially exceeds between-trial sd, a higher episode count would tighten my estimates; where it does not, five episodes is sufficient.

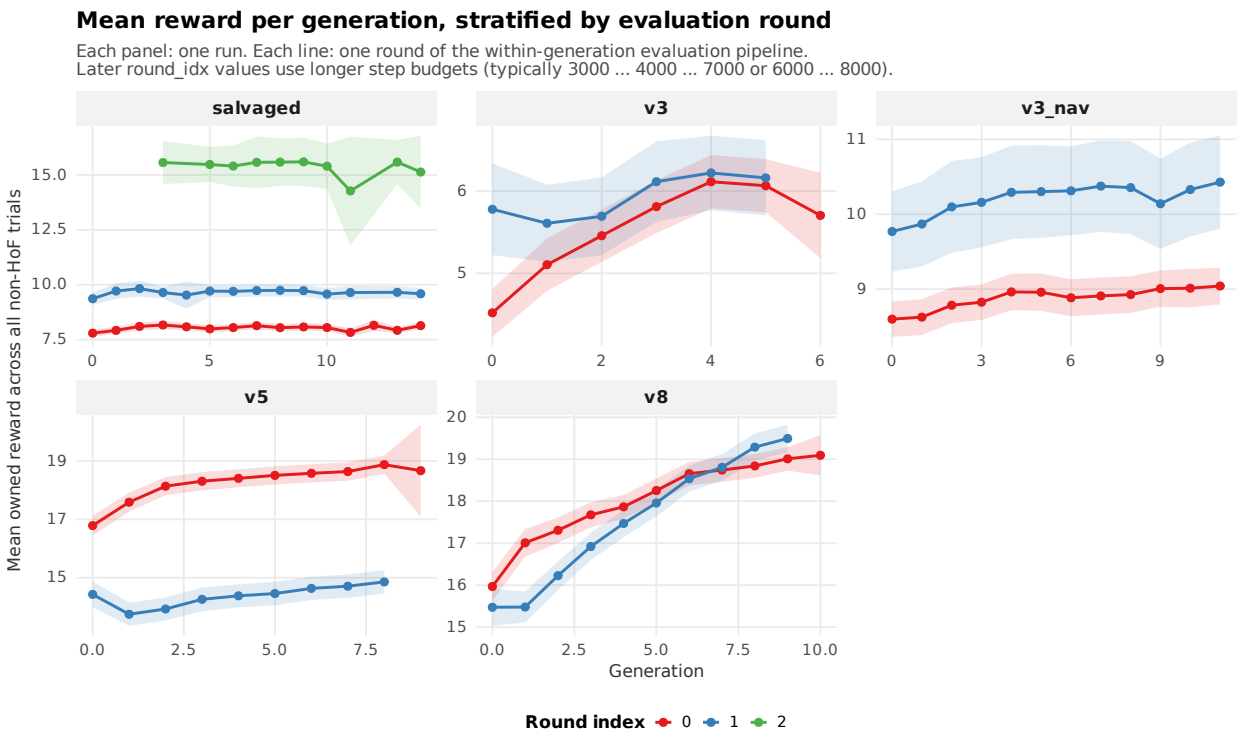



**Figure 12:** Mean reward per generation, stratified by evaluation round within the generation. In the successful runs, mean reward improves at every round, not just the final selected round, confirming that the GA is improving the underlying population rather than merely narrowing selection.