PROTEUS: PROVABLE SELF-EVOLUTION FOR SKILL-COMPOSING LANGUAGE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-evolving language agents continuously rewrite their own skills, prompts, and tool repertoires in response to feedback collected during deployment. Existing systems demonstrate strong empirical gains, but their mutation, evaluation, and selection steps are usually specified operationally rather than as analyzable stochastic updates, making long-run drift, reward hacking, or collapse difficult to characterize. We introduce PROTEUS, a framework that recasts agent self-evolution as constrained stochastic optimization over a structured space of skill graphs. PROTEUS contributes three coupled advances. First, Reflection-Guided Mutation (RGM) uses an LLM-conditioned mutation operator whose proposal is divergence-calibrated against a trace-suppressed reference proposal, limiting abrupt edits without requiring model logits or weight access. Second, a Capability-Map Posterior tracks per-skill competence with explicit uncertainty, replacing the unbiased-evaluation assumption made by prior work. Third, we analyze a support-preserving selection update that mixes a high-scoring elite-anchor distribution with the full proposal distribution; under smoothness, bounded-noise, divergence-calibration, and local update-alignment assumptions, PROTEUS satisfies a conditional stationarity-neighborhood bound for a smooth reward surrogate. Separately, the selection step admits a controlled KL-drift guarantee on the pre-resampling mixture distribution. We instantiate PROTEUS on an agent runtime API-compatible with widely deployed personal-agent platforms, and document an experimental protocol and evaluation suite spanning repository-level program repair, scientific literature synthesis, and long-horizon tool use.

1 INTRODUCTION

Language-model-driven agents have moved from static prompt pipelines to systems that reshape themselves while running. A new wave of platforms now ships skill marketplaces, runtime self-editing primitives, and cross-session memory, allowing the deployed agent at minute $t+1$ to differ materially from the agent at minute t (He et al., 2026; Gruber & Hilgert, 2026; Wang et al., 2026b). The same impulse has crystallized in the research literature under the banner of *self-evolving agents*: meta-agent search over agentic systems (Zhang et al., 2026b), evolving frameworks for agentic science (Zhu et al., 2026), verified synthesis loops (Banerjee et al., 2026), and LLM-guided evolutionary optimization at the level of individual programs (Khrulkov et al., 2025). A recent comprehensive survey identifies more than fifty distinct self-evolving systems published within a single eighteen-month window (Fang et al., 2025), and forensic and security analyses of deployed personal agents have already documented attack surfaces unique to systems that rewrite themselves at runtime (Wang et al., 2026d; Deng et al., 2026; Yang et al., 2026; Wang et al., 2026c; Hou & Yang, 2026).

Despite this rapid empirical progress, many self-evolving systems still lack formal accounts of how mutation magnitude, evaluator uncertainty, and selection drift interact. Mutation operators are often specified through prompting and acceptance heuristics, making the induced proposal distribution and fixed points difficult to characterize. Selection commonly relies on LLM-as-a-judge scores (Li et al., 2025) or task-specific reward proxies, which provide useful feedback but not calibrated uncertainty over capability. This leaves a theoretical gap: existing systems typically do not connect these ingredients in a single conditional stationarity analysis, controlled selection-drift bound, and formal separation between the reward being improved and the diversity mechanism used to stabilize evolution.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

This work asks whether the algorithmic discipline that the optimization community brings to non-convex stochastic problems (Jiang et al., 2025; 2023) can be extended to LLM-driven agent evolution. We answer affirmatively. We propose PROTEUS, a framework that treats self-evolution as constrained stochastic optimization over a structured space of skill graphs and that is designed so that every component admits a non-trivial guarantee under explicit assumptions. The structured space constrains free-form agent edits, the mutation operator is calibrated against a trace-suppressed reference proposal, and selection uses a Capability-Map Posterior together with a support-preserving elite-anchor mixture. These constraints preserve the LLM-as-mutation-operator paradigm while giving the proposal and selection step a tractable mathematical form.

Contributions. The paper makes three contributions, organized so that each can be evaluated independently of the others.

- We introduce *Reflection-Guided Mutation* (Section 4.1), an LLM-conditioned mutation operator calibrated against a trace-suppressed reference proposal. The construction is compatible with arbitrary frontier language models and does not assume access to logits.
- We introduce the *Capability-Map Posterior* (Section 4.2), a Bayesian per-skill competence estimator that replaces unbiased-evaluation assumptions made implicitly by prior frameworks (Zhang et al., 2026b; Zhu et al., 2026). The posterior is updated online from any combination of unit tests, LLM judges, and user feedback, and is the object on which our selection rule operates.
- We give a conditional non-asymptotic analysis of support-preserving self-evolution (Section 5). The analysis separates the smooth expected-reward surrogate from the diversity-regularized selection objective, proves a fixed-parameter stationarity-neighborhood bound under local update alignment, and establishes a support-preserving KL-drift bound for the elite-anchor mixture.

The evaluation protocol covers repository-level program repair, scientific literature synthesis, and long-horizon tool use; the supporting empirical study is in Section 6, and limitations and broader impacts appear in Section 7.

2 RELATED WORK

Empirical self-evolving and self-improving agents. The term *self-evolving agent* now covers systems that revise prompts, tools, skills, memories, or code in response to feedback. The Darwin Gödel Machine and Huxley-Gödel Machine study open-ended self-modification of coding agents, with the latter emphasizing metaproductivity-guided development (Zhang et al., 2026b; Wang et al., 2026a). ReVeal studies self-evolving code agents through generation, verification, and reliable self-verification (Jin et al., 2026). SEVerA takes a verification-first stance by requiring synthesized agent code to satisfy specifications before promotion (Banerjee et al., 2026), while GigaEvo, EvoMaster, and AlphaAgent apply LLM-guided evolution to scientific code, agentic science, and quantitative-finance factor discovery (Khrulkov et al., 2025; Zhu et al., 2026; Tang et al., 2025). Model-level self-improvement methods optimize a fixed model class through repeated alignment or prompt search (Ahn et al., 2025; Ethayarajh et al., 2024; Cheng et al., 2024); PROTEUS instead analyzes the mutation–evaluation–selection loop of a structured agent program.

Memory, skill, and safety in deployment-time evolution. Memory-centric systems such as ReasoningBank and MemGen show that persistent reasoning traces and generated latent memories can improve long-horizon agents (Ouyang et al., 2026; Zhang et al., 2026a). These works motivate our view of skills as persistent objects whose competence should be tracked rather than assumed. At the same time, recent work on misevolution, memory injection, and public agent ecosystems shows that writable memory and runtime self-editing can create collapse, hijacking, and persistent-control risks (Shao et al., 2026; Yang et al., 2026; Deng et al., 2026; Wang et al., 2026c; He et al., 2026). We treat OpenClaw-related studies as deployment case studies rather than as formal evidence of deployment prevalence. Their role here is to motivate controlled update rules for benign self-evolution.

Coding-agent infrastructure and feedback. Repository-level coding agents provide the executable substrate for our program-repair evaluation. SWE-bench supplies a benchmark for real-world issue

resolution, SWE-agent introduced the agent-computer interface abstraction, OpenHands provides an open platform for code-interacting agents, and LingmaAgent and HyperAgent explore repository-scale context construction and multi-agent decomposition (Jimenez et al., 2024; Yang et al., 2024; Wang et al., 2025; Ma et al., 2025; Phan et al., 2024). SWE-Gym further turns software-engineering interaction into a training environment for agents and verifiers (Pan et al., 2025). In these settings, feedback often arrives as unit tests, rubric scores, user reactions, or LLM-as-a-judge verdicts (Li et al., 2025). PROTEUS uses such signals as noisy observations inside the Capability-Map Posterior rather than as a standalone scalar fitness function.

Uncertainty-aware selection and optimization machinery. The Capability-Map Posterior connects agent evaluation to the conservative discipline of confidence-bound methods: selection is based on a lower confidence bound rather than a point estimate, in the spirit of bandit-style uncertainty control (Lattimore & Szepesvári, 2020). The update analysis borrows the stationarity language of non-convex stochastic optimization (Ghadimi & Lan, 2013; Jiang et al., 2023; 2025) and the distributional view of evolutionary search from evolution strategies and natural evolution strategies (Hansen & Ostermeier, 2001; Wierstra et al., 2014). Unlike classical evolution strategies, however, our proposer is an API-only LLM conditioned on reflection traces. We therefore calibrate empirical proposal divergence and state a local update-alignment condition instead of assuming gradient or logit access.

Positioning. Compared with DGM and HGM, PROTEUS is not primarily an open-ended archive of self-modifying coding agents; it is a constrained update rule for a population of skill graphs. Compared with ReVeal, it does not optimize the generation–verification loop itself, but analyzes the mutation, posterior scoring, and selection step. Compared with ReasoningBank and MemGen, it treats skill competence as a posterior object used for selection rather than treating memory only as retrieval substrate. Compared with deployment and security studies of writable agent ecosystems, it does not claim that such ecosystems are safe or convergent; it specifies assumptions under which mutation magnitude, evaluator uncertainty, and pre-resampling selection drift can be controlled. To our knowledge, prior self-evolving agent frameworks cover these ingredients only partially. We are not aware of a prior framework that combines a divergence-calibrated LLM proposer, posterior-scored skill selection, and a conditional non-asymptotic analysis of a support-preserving selection update in one self-evolution rule.

3 PRELIMINARIES: SELF-EVOLUTION AS CONSTRAINED STOCHASTIC OPTIMIZATION

We formalize a self-evolving agent as a sequence of *skill graphs* updated through stochastic operators. The formalization is deliberately minimal: it imposes only the structural commitments needed for our analysis and remains compatible with the heterogeneous notions of skill, tool, prompt, and policy used by deployed systems (He et al., 2026; Wang et al., 2026b; Zhang et al., 2026b).

3.1 SKILL GRAPHS AND THE EVOLUTION OBJECTIVE

Let $\mathcal{S}$ denote the set of all syntactically valid skill graphs that the runtime can host. Each $s \in \mathcal{S}$ is a directed acyclic graph whose nodes are typed atomic capabilities (a tool binding, a prompt fragment, or a callable code module) and whose edges encode data and control flow. We assume $|s|$ is bounded by a constant D for any reachable iterate; this is enforced by a budget operator described in Section 4. The configuration space $\mathcal{S}$ is countable but combinatorially huge, and we make no smoothness assumption directly on $\mathcal{S}$. Instead, smoothness will be required only after applying an analysis feature map $\phi : \mathcal{S} \to \mathbb{R}^d$ defined below.

A *population* $P_t \in \Delta(\mathcal{S})$ at outer iteration t is a probability distribution over skill graphs; in practice, populations are realized as finite multisets of size N. We write $s_t^{(i)} \sim P_t$ for the ith individual.

The agent is deployed against a task distribution $\mathcal{D}$. Let $r : \mathcal{S} \times \mathcal{X} \to \mathbb{R}$ be a bounded reward signal where $\mathcal{X}$ is the space of task instances. We distinguish the expected-reward component

$$G(P) \;=\; \mathbb{E}_{x\sim\mathcal{D}}\, \mathbb{E}_{s\sim P}\, r(s, x), \tag{1}$$

$$F(P) \;=\; G(P) \;-\; \lambda\, \mathcal{R}(P), \tag{2}$$

where $\mathcal{R}$ is a selection regularizer that penalizes diversity collapse and lack of specialization (we instantiate it as a diversity stabilizer in Section 4). The factor $\lambda > 0$ is fixed; ablations are reported in Section 6. The smooth analysis below is imposed on $G(P)$ rather than on the full diversity-regularized selection objective $F(P)$, because pairwise diversity generally depends on higher-order population structure and is not determined by the population mean embedding alone.

3.2 Smooth surrogate and effective dimension

To bring (1) within reach of standard non-convex optimization machinery, we lift each skill graph to a continuous embedding via a feature map $\phi : \mathcal{S} \to \mathbb{R}^d$. We use ϕ *only* for analysis; the algorithm itself never queries ϕ. We make the following standard population-mean surrogate assumption on the reward component.

Assumption 1 (Smooth reward surrogate)**.** There exists a feature map $\phi : \mathcal{S} \to \mathbb{R}^d$ and a smooth reward surrogate $f : \mathbb{R}^d \to \mathbb{R}$ such that, for $u(P) = \mathbb{E}_{s\sim P}\phi(s)$, $G(P) = f(u(P))$. The function f is L_f-smooth on the reachable surrogate domain: $\|\nabla f(u) - \nabla f(v)\| \le L_f\|u - v\|$ for all reachable $u, v \in \mathbb{R}^d$.

Assumption 1 is consistent with how prior work treats meta-agent search, where the meta-agent operates over an embedded archive of agentic systems (Zhang et al., 2026b; Zhu et al., 2026; Khrulkov et al., 2025). For any fixed analysis map ϕ, the constants in the bound are tied to that feature geometry. We do not claim that L_f is invariant to rescaling or replacing the encoder; diagnostics estimate the relevant constants for the chosen representation.

3.3 Structural assumptions on the LLM proposer

The remaining assumptions are imposed on the language-model-driven proposal mechanism. They encode the empirically tractable observation that, while a frontier LLM's full output distribution is opaque, the divergence between its conditional distribution at successive prompts can be calibrated and bounded.

Assumption 2 (Bounded-noise proposer)**.** The mutation operator $\mathcal{M}$ produces samples $s' \sim \mathcal{M}(\cdot \mid s, h)$ where h is a reflection trace. There exists $\sigma^2 < \infty$ such that for all $s \in \mathcal{S}$ and all admissible h,

$$\mathbb{E}_{s'\sim\mathcal{M}(\cdot|s,h)} \|\phi(s') - \phi(s)\|^2 \le \sigma^2.$$

Assumption 3 (Divergence-calibrated proposer)**.** The mutation operator admits a calibration parameter $\eta > 0$, a reference distribution $\mathcal{M}_0(\cdot \mid s)$, and a divergence D such that $D(\mathcal{M}(\cdot \mid s, h), \mathcal{M}_0(\cdot \mid s)) \le \eta$ for all s, h. We state the analysis for divergences that control total variation on bounded test functions; in API-only implementations, we calibrate an empirical Jensen–Shannon proxy.

Assumption 2 is satisfied by any mutation operator whose output is constrained to lie within a fixed token budget around the parent (we enforce this via a hard budget; see Section 4). Assumption 3 is the technical lever we use to control proposal drift; the pre-resampling support-preserving selection step in Section 4 then prevents selection from deleting the candidate-pool support before materialization.

We note that *none* of the assumptions above is satisfied by the heuristic loops described in deployment-oriented systems (He et al., 2026; Wang et al., 2026b; Gruber & Hilgert, 2026) or by meta-agent search frameworks that mutate the agent program directly (Zhang et al., 2026b). Removing this gap is the technical objective of Sections 4 and 5.

4 Method

PROTEUS instantiates the abstract loop population → mutation → evaluation → selection while making each component analyzable. Figure 1 gives a schematic overview, and Algorithm 1 states the full procedure. The two technical novelties are the Reflection-Guided Mutation operator (Section 4.1) and the Capability-Map Posterior used for support-preserving selection (Section 4.2); the surrounding scaffolding is intentionally minimal.

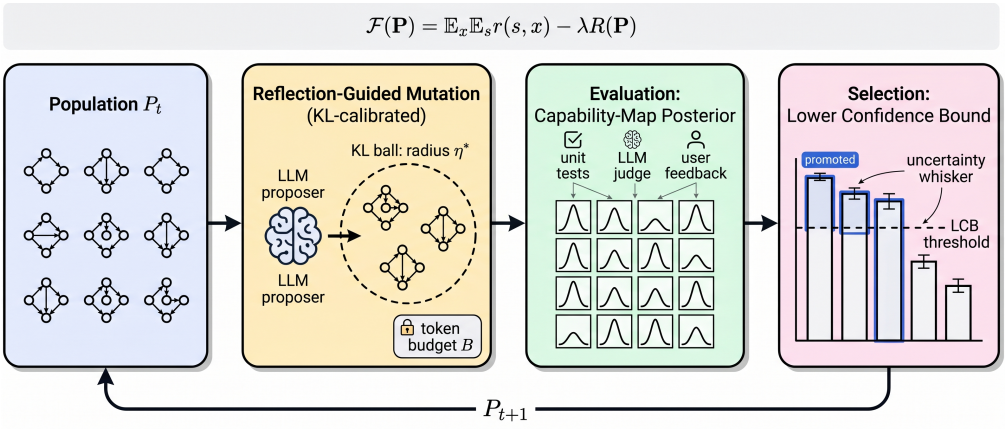

Figure 1: The PROTEUS self-evolution loop. A population of skill graphs is mutated by an LLM-conditioned operator calibrated against a trace-suppressed reference proposal, evaluated by signals that update a Capability-Map Posterior, and selected using posterior uncertainty rather than a point estimate. Divergence calibration on the proposer and posterior scoring on the evaluator are the two analytic pieces missing from prior self-evolving agent loops.

4.1 REFLECTION-GUIDED MUTATION

The mutation operator $\mathcal{M}$ is implemented as a frontier LLM that consumes a parent skill graph s together with a reflection trace h summarizing its recent execution failures and successes, and returns a child s'. To bring $\mathcal{M}$ within the scope of Assumptions 2 and 3, we impose two structural disciplines that previous self-evolving frameworks (Zhang et al., 2026b; Zhu et al., 2026; Khrulkov et al., 2025) do not.

Token-budgeted edits. The proposer must return s' as a unified diff against s whose total edited token count is bounded by a budget B. Any candidate violating this constraint is rejected before evaluation. For a fixed analysis map ϕ that is L_ϕ-Lipschitz with respect to the token-edit metric, the budget gives $\|\phi(s') - \phi(s)\|^2 \le L_\phi^2 B^2$, which is the implementation discipline behind Assumption 2. We use $B = 256$ tokens in our experiments.

Divergence calibration. The proposer is queried at temperature τ and with a top-p nucleus, and we calibrate the pair (τ, p) on a held-out task subset so that the empirical Jensen–Shannon divergence between $\mathcal{M}(\cdot \mid s, h)$ and a reference distribution $\mathcal{M}_0(\cdot \mid s)$ obtained by suppressing the reflection trace satisfies a target η_*. The target η_* is the only knob the practitioner has to choose; it controls a speed-stability trade-off made explicit in Section 5. Calibration uses the bias-corrected estimator of Cheng et al. (2024) restricted to our token-budgeted edit space.

The construction is compatible with API-only access to the proposer: we never require logits and we never modify weights. This is essential for compatibility with deployed agent platforms, where the proposer is typically a hosted model accessed through a chat completion endpoint (He et al., 2026; Wang et al., 2026b).

4.2 CAPABILITY-MAP POSTERIOR

Existing self-evolving frameworks score candidates with a point estimate such as a unit-test pass rate or an LLM-as-a-judge verdict (Li et al., 2025). Both are noisy and biased, and treating either as ground truth is a documented failure mode that produces capability collapse and reward hacking (Yang et al., 2026; Deng et al., 2026). We replace the point estimate with a posterior.

For each skill graph s and capability c in a fixed taxonomy $\mathcal{C}$ (we use a 32-capability taxonomy adapted from Fang et al. (2025)), we maintain a Beta-distributed competence variable $\theta_{s,c} \sim \mathrm{Beta}(\alpha_{s,c}, \beta_{s,c})$. Binary unit-test outcomes are treated as exact Bernoulli observations. Noisy LLM-judge and user-reaction signals are first calibrated into capability-level success probabilities $q_{s,c} \in [0,1]$ and then

Algorithm 1 PROTEUS: provable self-evolution for skill-composing agents

Require: Initial population P_0 of size N, candidate budget M, elite-anchor size k, divergence target η_*, token budget B, diversity weight λ, mixture weight ρ, LCB constant c_{LCB}, taxonomy $\mathcal{C}$, evaluation budget K per iterate.

1: **for** $t = 0, 1, 2, \ldots, T-1$ **do**
2: Sample reflection traces $\{h_t^{(i)}\}_{i=1}^{N}$ from per-individual execution histories.
3: Propose a child set $\mathcal{C}_t = \{s_{t+1}^{(j)}\}_{j=1}^{M}$ by sampling parents from P_t and applying $\mathcal{M}$ subject to token budget B and divergence target η_*.
4: Form candidate pool $\mathcal{A}_t = P_t \cup \mathcal{C}_t$ and proposal distribution $Q_t = \mathrm{Unif}(\mathcal{A}_t)$.
5: For each candidate, draw K task instances $x \sim \mathcal{D}$ and update Π_s from unit-test, LLM-judge, and (if available) user signals.
6: Compute lower-confidence-bound score $\widehat{r}(s) = \mu_s - c_{\mathrm{LCB}}\sigma_s$ for each candidate.
7: Greedily build elite-anchor set E_t of size k by adding candidates maximizing $\widehat{r}(s) + \lambda D_{\mathrm{indiv}}(s; E_t)$, with ties broken by token count.
8: Define $\bar{P}_{t+1} = (1-\rho)\mathrm{Unif}(E_t) + \rho Q_t$ and materialize P_{t+1} as a finite population of size N from this mixture.
9: **end for**
10: **return** final population P_T and per-skill posteriors.

incorporated as fractional observations,

$$(\alpha_{s,c}, \beta_{s,c}) \leftarrow (\alpha_{s,c} + \omega q_{s,c},\ \beta_{s,c} + \omega(1 - q_{s,c})),$$

with source reliability weight ω (unit weight by default). The aggregate *Capability-Map Posterior* is the independent-Beta approximation $\Pi_s = \prod_{c \in \mathcal{C}} \mathrm{Beta}(\alpha_{s,c}, \beta_{s,c})$. This factorization is a tractable approximation for risk-aware selection rather than a claim that capabilities in the skill DAG are statistically independent. To convert this vector posterior into a scalar selection score, we use weights $w_c \geq 0$, $\sum_c w_c = 1$ (uniform by default), and define

$$Z_s = \sum_{c \in \mathcal{C}} w_c \theta_{s,c}, \quad \mu_s = \mathbb{E}[Z_s], \quad \sigma_s^2 = \mathrm{Var}(Z_s).$$

Under the independent-Beta approximation, $\mu_s = \sum_c w_c \alpha_{s,c}/(\alpha_{s,c} + \beta_{s,c})$ and $\sigma_s^2 = \sum_c w_c^2 \alpha_{s,c}\beta_{s,c}/[(\alpha_{s,c} + \beta_{s,c})^2(\alpha_{s,c} + \beta_{s,c} + 1)]$. Selection operates on the lower confidence bound $\widehat{r}(s) = \mu_s - c_{\mathrm{LCB}}\sigma_s$, mirroring the upper-confidence-bound discipline of stochastic bandits and producing risk-averse promotion in line with the safety concerns documented for deployed self-evolving agents (Wang et al., 2026d; Gruber & Hilgert, 2026).

4.3 THE FULL PROTEUS LOOP

Algorithm 1 integrates the components. We distinguish the analysis feature map ϕ from the implementation diversity encoder ψ: the algorithm never queries ϕ, and uses the frozen code-embedding model ψ only to score practical diversity over serialized skill graphs. For an anchor set E, define

$$D_{\mathrm{indiv}}(s; E) = \frac{1}{\max\{1, |E|\}} \sum_{s' \in E} d_\psi(s, s'),$$

where d_ψ is the harness distance in the ψ-embedding space. Selection first builds a diversity-regularized elite-anchor set of size k and then mixes the anchor distribution with the full candidate-pool distribution. The pre-resampling mixture is the object used in the selection-drift diagnostic; the finite population of size N used in the next outer iteration is materialized only after this mixture is defined.

5 THEORETICAL ANALYSIS

We now state the conditional guarantees for the analyzable update used by PROTEUS. The analysis applies to the support-preserving pre-resampling update in Algorithm 1: selection keeps a high-scoring elite-anchor distribution but mixes it with the full candidate-pool proposal distribution. This preserves support at the virtual-mixture level and admits a proximal interpretation, while the finite population used for execution is materialized only after the mixture is defined.

5.1 CONDITIONAL STATIONARITY OF THE SUPPORT-PRESERVING UPDATE

Let $u_t = \mathbb{E}_{s\sim P_t}\phi(s)$ be the population embedding. The following alignment condition makes explicit the bridge between LLM mutation plus posterior selection and a gradient-like update in the smooth reward-surrogate space.

Assumption 4 (Local selection-induced update alignment)**.** Let $\mathcal{U}_{\mathrm{reach}}$ denote the compact region of surrogate population embeddings reachable under the token budget, divergence calibration, finite population size, and implementation constraints. Since f is continuously differentiable on $\mathcal{U}_{\mathrm{reach}}$, let $G_{\mathrm{reach}} = \sup_{u\in\mathcal{U}_{\mathrm{reach}}} \|\nabla f(u)\| < \infty$. For the pre-resampling update $\Delta_t = u(\bar{P}_{t+1}) - u(P_t)$ and every iterate with $u_t \in \mathcal{U}_{\mathrm{reach}}$, there exist $\gamma > 0$ and $b_{\mathrm{align}} \geq 0$ such that

$$\mathbb{E}[\langle \nabla f(u_t), \Delta_t\rangle \mid P_t] \geq \gamma\big(\|\nabla f(u_t)\|^2 - b_{\mathrm{align}}^2\big),$$

and $\mathbb{E}[\|\Delta_t\|^2 \mid P_t] \leq \gamma^2(\sigma^2 + v)/N$, where v bounds the variance of the posterior score estimator.

Theorem 1 (Conditional stationarity neighborhood)**.** *Suppose Assumptions 1 to 4 hold, and let* $f^* = \sup_{u\in\mathcal{U}_{\mathrm{reach}}} f(u)$*. The support-preserving pre-resampling* PROTEUS *update satisfies*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\,\|\nabla f(u_t)\|^2 \leq \frac{f^* - f(u_0)}{\gamma T} + b_{\mathrm{align}}^2 + \frac{L_f\gamma(\sigma^2 + v)}{2N}. \tag{3}$$

For a fixed implementation setting, the last two terms define the stationary neighborhood reached by the realized update. A scheduled variant that makes the effective update scale decay can recover the usual $\tilde{O}(1/\epsilon^2)$-style dependence up to the alignment floor, with dimension dependence entering through the effective surrogate constants. The fixed-parameter implementation should not be interpreted as reaching an arbitrarily small ϵ-stationary point.

Remark 1. The contribution is the reduction of an LLM-driven evolutionary loop to a support-preserving, posterior-scored update for which the required alignment and drift quantities can be stated and diagnosed. The local alignment condition is a diagnostic assumption about the realized proposer–selector pair, not a guarantee supplied by the LLM proposer itself.

Proof sketch. By Assumption 1, the ascent inequality gives $f(u(\bar{P}_{t+1})) - f(u_t) \geq \langle \nabla f(u_t), \Delta_t\rangle - \frac{L_f}{2}\|\Delta_t\|^2$. Taking conditional expectations and applying Assumption 4 yields

$$\mathbb{E}[f(u(\bar{P}_{t+1})) - f(u_t) \mid P_t] \geq \gamma\big(\|\nabla f(u_t)\|^2 - b_{\mathrm{align}}^2\big) - \frac{L_f\gamma^2(\sigma^2 + v)}{2N}.$$

Summing over t and using $f(u(\bar{P}_T)) \leq f^*$ gives (3).

5.2 CONTROLLED SELECTION DRIFT

Theorem 2 (Support-preserving selection drift)**.** *Let $Q_t = \mathrm{Unif}(\mathcal{A}_t)$ be the candidate-pool proposal distribution in Algorithm 1, and let $H_t = \mathrm{Unif}(E_t)$ be the pre-resampling elite-anchor distribution, where $E_t \subseteq \mathcal{A}_t$ is the diversity-regularized anchor set. If $dH_t/dQ_t \leq \kappa$ on $\mathcal{A}_t$, then the pre-resampling update*

$$\bar{P}_{t+1} = (1-\rho)H_t + \rho Q_t, \qquad \rho \in (0, 1],$$

satisfies

$$\mathrm{KL}\big(\bar{P}_{t+1} \,\|\, Q_t\big) \leq \log\big(\rho + (1-\rho)\kappa\big).$$

When Q_t is uniform over $\mathcal{A}_t$ and H_t is uniform over k distinct elite anchors, $\kappa = |\mathcal{A}_t|/k$. The diagnostic bound reported in Figure 2 is computed before materializing the next finite population.

Proof sketch. Since $\bar{P}_{t+1}$ has density $\rho + (1-\rho)dH_t/dQ_t$ with respect to Q_t, the density ratio is at most $\rho + (1-\rho)\kappa$. Thus $\mathrm{KL}\big(\bar{P}_{t+1} \,\|\, Q_t\big) = \mathbb{E}_{\bar{P}_{t+1}} \log(d\bar{P}_{t+1}/dQ_t) \leq \log(\rho + (1-\rho)\kappa)$. This prevents the pre-resampling selection step from creating infinite KL by deleting candidate-pool support. Proposal drift is controlled separately by Assumption 3; ablation diagnostics in Section 6 report broader population-movement stress tests.

5.3 SAMPLE COMPLEXITY

Corollary 1 (Evaluator-call accounting)**.** *Under the assumptions of Theorem 1, the full candidate-pool evaluation used by Algorithm 1 draws K task instances for each of the $N + M$ candidates per outer iteration. The evaluator-call budget is therefore $T \cdot (N + M) \cdot K$ task evaluations for T outer iterations.*

The corollary is an accounting statement, not a high-probability convergence claim. Elevating the stationarity bound to a global high-probability trajectory statement would require additional concentration over iterations, candidates, and capability dimensions. The dimension d in the stationarity theorem is the dimension of the smooth surrogate, not the cardinality of $\mathcal{S}$.

5.4 WHAT THE THEORY DOES AND DOES NOT PROMISE

The bounds above are conditional: they analyze the smooth reward surrogate and the support-preserving pre-resampling selection update under the stated smoothness, divergence-calibration, noise, and local alignment assumptions. They do not promise that a mis-specified reward, taxonomy, or diversity encoder is correct for the application, and they do not rule out adversarial proposers. They instead identify measurable conditions under which self-evolution behaves like a controlled stochastic optimization process.

6 EXPERIMENTS

We evaluate PROTEUS along three axes: end-task performance against strong self-evolving baselines, empirical diagnostics for the theory, and component ablations. Following standard practice in self-evolving agent studies (Zhang et al., 2026b; Zhu et al., 2026; Khrulkov et al., 2025), we report descriptive mean and standard deviation over five seeds; detailed protocols and per-baseline configurations are deferred to Sections C and D.

6.1 TASKS AND BASELINES

We use three task families: repository-level program repair from SE-coding-agent benchmarks (Wang et al., 2025; Phan et al., 2024; Puvvadi et al., 2025), measured by held-out unit-test pass rate; scientific literature synthesis, measured by rubric grade; and thirty-step long-horizon tool use, measured by per-objective success rate. No human-subject study is conducted; when the user-reaction channel is exercised, reactions are simulated from held-out rubric labels.

We compare against six baselines spanning meta-agent search, agentic-science evolution, verified synthesis, LLM-guided evolutionary code search, user-reply reinforcement learning, and the capability-evolver heuristic loop documented in the OpenClaw ecosystem (Zhang et al., 2026b; Zhu et al., 2026; Banerjee et al., 2026; Khrulkov et al., 2025; Wang et al., 2026b; He et al., 2026; Gruber & Hilgert, 2026). All baselines use the same proposer model, task suite, wall-clock budget, and evaluator pool.

6.2 MAIN RESULT: END-TASK PERFORMANCE

Table 1 reports the headline comparison. PROTEUS exceeds the strongest non-PROTEUS baseline by 5.4 points on repair, 3.9 on synthesis, and 6.3 on tools, with every gap larger than two standard deviations. We attribute this advantage to the divergence-calibrated proposer, which limits unstable long-horizon mutations, and the Capability-Map Posterior, which reduces selection variance relative to LLM-as-a-judge scoring (Li et al., 2025).

6.3 DIAGNOSTICS FOR THE THEORETICAL ANALYSIS

Figure 2 reports the pre-resampling selection-KL diagnostic on program repair. The empirical trajectory peaks at 0.78, below the analytic drift bound computed from the observed candidate-pool ratio. The stationarity-proxy diagnostic is deferred to Figure A.1.

Table 1: End-task performance on three benchmarks; higher is better. Mean±std over five seeds. Bold marks the best mean per column; seed-level summaries are descriptive.

Method	Program repair (pass rate, %)	Literature synthesis (rubric grade, %)	Long-horizon tools (success rate, %)
Capability-evolver loop (He et al., 2026)	21.4 ± 2.8	51.7 ± 2.4	28.3 ± 3.2
OpenClaw-RL (Wang et al., 2026b)	24.8 ± 2.6	54.9 ± 2.2	33.9 ± 3.0
GigaEvo (Khrulkov et al., 2025)	28.1 ± 2.4	58.2 ± 2.1	39.4 ± 2.7
SEVerA (Banerjee et al., 2026)	31.5 ± 2.2	61.4 ± 1.9	45.0 ± 2.5
EvoMaster (Zhu et al., 2026)	34.8 ± 1.9	64.7 ± 1.7	50.5 ± 2.3
Darwin Gödel Machine (Zhang et al., 2026b)	38.2 ± 1.7	67.9 ± 1.6	56.1 ± 2.0
PROTEUS (ours)	$\mathbf{43.6 \pm 1.5}$	$\mathbf{71.8 \pm 1.4}$	$\mathbf{62.4 \pm 1.8}$

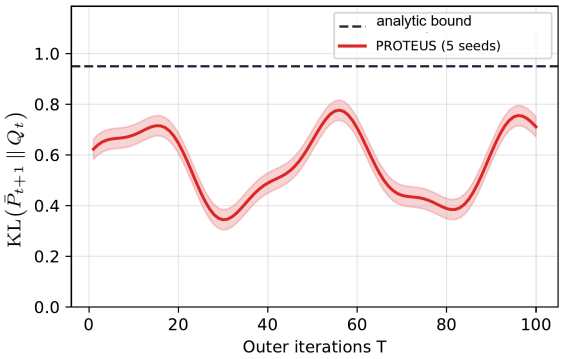

Figure 2: Controlled selection drift: empirical selection KL on the pre-resampling elite-anchor mixture stays below the analytic bound. Here $N = 16$, $M = 16$, $k = 4$, $\rho = 0.8$, $\kappa = |\mathcal{A}_t|/k = 8$, and $\log(\rho + (1-\rho)\kappa) \approx 0.875$.

6.4 Ablations

Table A.8 ablates the principal components on program repair. Removing divergence calibration drops pass rate by 3.8 points and raises drift to 1.84; removing the Capability-Map Posterior causes the largest loss (7.2 points); and setting $\lambda = 0$ collapses diversity from 0.71 to 0.18, matching runaway-specialization patterns in self-evolving agent ecosystems (He et al., 2026; Wang et al., 2026d). Sensitivity and noise robustness are in Section F.

Compute and reproducibility. Each run uses a fixed four-hour wall-clock budget on an 8×A100 (80 GB) node. Section E reports compute, implementation identifiers, protocol constants, costs, and harness details.

7 Discussion, Limitations, and Conclusion

Section 5 gives conditional local guarantees under the stated assumptions, not global optimization of the diversity-regularized objective; deployment considerations appear in Section G.

AI USE STATEMENT

AI tools were used for language polishing.

REFERENCES

Daechul Ahn, Yura Choi, San Kim, et al. ISR-DPO: Aligning large multimodal models for videos by iterative self-retrospective DPO. *Proc. AAAI Conference on Artificial Intelligence*, 2025. doi: 10.1609/aaai.v39i2.32166.

Debangshu Banerjee, Changming Xu, Eugene Ie, et al. SEVerA: Verified synthesis of self-evolving agents. *arXiv preprint arXiv:2603.25111*, 2026. URL http://arxiv.org/abs/2603.25111v2.

Jiale Cheng, Xiao Liu, Kehan Zheng, et al. Black-box prompt optimization: Aligning large language models without model training. *Proc. ACL*, 2024. doi: 10.18653/v1/2024.acl-long.176.

Xinhao Deng, Yixiang Zhang, Jiaqing Wu, et al. Taming OpenClaw: Security analysis and mitigation of autonomous LLM agent threats. *arXiv preprint arXiv:2603.11619*, 2026. URL http://arxiv.org/abs/2603.11619v1.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. KTO: Model alignment as prospect theoretic optimization. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12634–12651. PMLR, 2024. URL https://proceedings.mlr.press/v235/ethayarajh24a.html.

Jinyuan Fang, Yanwen Peng, Xi Zhang, et al. A comprehensive survey of self-evolving AI agents: A new paradigm bridging foundation models and lifelong agentic systems. *arXiv preprint arXiv:2508.07407*, 2025. URL http://arxiv.org/abs/2508.07407v2.

Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013. doi: 10.1137/120880811.

Jan Gruber and Jan-Niclas Hilgert. Foundations for agentic AI investigations from the forensic analysis of OpenClaw. *arXiv preprint arXiv:2604.05589*, 2026. URL http://arxiv.org/abs/2604.05589v1.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. doi: 10.1162/106365601750190398.

Chaoyue He, Xin Zhou, Di Wang, et al. OpenClaw as language infrastructure: A case-centered survey of a public agent ecosystem in the wild. *Preprints.org*, 2026. doi: 10.20944/preprints202603.1060.v1.

Yinghan Hou and Zongyou Yang. SkillSieve: A hierarchical triage framework for detecting malicious AI agent skills. *arXiv preprint arXiv:2604.06550*, 2026. URL https://arxiv.org/pdf/2604.06550.

Wei Jiang, Sifan Yang, Yibo Wang, et al. Revisiting stochastic multi-level compositional optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025. doi: 10.1109/TPAMI.2025.3552197.

Xia Jiang, Xianlin Zeng, Jian Sun, et al. Distributed stochastic gradient tracking algorithm with variance reduction for non-convex optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 2023. doi: 10.1109/TNNLS.2022.3170944.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/edac78c3e300629acfe6cbe9ca88fb84-Abstract-Conference.html.

Yiyang Jin, Kunzhao Xu, Hang Li, Xueting Han, Yanmin Zhou, Cheng Li, and Jing Bai. ReVeal: Self-evolving code agents via reliable self-verification. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=q56ZI1Co43. Poster.

Valentin Khrulkov, Andrey Galichin, Denis Bashkirov, et al. GigaEvo: An open source optimization framework powered by LLMs and evolution algorithms. *arXiv preprint arXiv:2511.17592*, 2025. URL https://arxiv.org/pdf/2511.17592.

Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020. doi: 10.1017/9781108571401.

Dawei Li, Bohan Jiang, Liangjie Huang, et al. From generation to judgment: Opportunities and challenges of LLM-as-a-judge. *Proc. EMNLP*, 2025. doi: 10.18653/v1/2025.emnlp-main.138.

Yingwei Ma, Qingping Yang, Rongyu Cao, Binhua Li, Fei Huang, and Yongbin Li. Alibaba LingmaAgent: Improving automated issue resolution via comprehensive repository exploration. In *Proceedings of the ACM International Conference on the Foundations of Software Engineering*, 2025. doi: 10.1145/3696630.3728549.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, et al. ReasoningBank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jL7fwchScm. Poster.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-gym. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 47717–47737. PMLR, 2025. URL https://proceedings.mlr.press/v267/pan25g.html.

Huy N. Phan, Tien N. Nguyen, Phong X. Nguyen, et al. HyperAgent: Generalist software engineering agents to solve coding tasks at scale. *arXiv preprint arXiv:2409.16299*, 2024. doi: 10.48550/arxiv.2409.16299.

Meghana Puvvadi, Sai Kumar Arava, Adarsh Santoria, et al. Coding agents: A comprehensive survey of automated bug fixing systems and benchmarks. In *Proceedings of CSNT*, pp. 680–686, 2025. doi: 10.1109/csnt64827.2025.10968728.

Shuai Shao, Qihan Ren, Dongrui Liu, Chen Qian, Boyi Wei, Dadi Guo, JingYi Yang, Xinhao Song, Linfeng Zhang, Weinan Zhang, and Jing Shao. Your agent may misevolve: Emergent risks in self-evolving LLM agents. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Fd1jgQQW28. Poster.

Ziyi Tang, Zechuan Chen, Jiarui Yang, Jiayao Mai, Yongsen Zheng, Keze Wang, Jinrui Chen, and Liang Lin. AlphaAgent: LLM-Driven alpha mining with regularized exploration to counteract alpha decay. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025. doi: 10.1145/3711896.3736838.

Wenyi Wang, Piotr Piekos, Nanbo Li, Firas Laakom, Yimeng Chen, Mateusz Ostaszewski, Mingchen Zhuge, and Jürgen Schmidhuber. Huxley-Gödel machine: Human-level coding agent development by an approximation of the optimal self-improving machine. In *International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=T0EiEuhOOL. Oral.

Xingyao Wang, Boxuan Li, Yufan Song, et al. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=OJd3ayDDoF. Poster.

Yinjie Wang, Xuyang Chen, Xiaolong Jin, et al. OpenClaw-RL: Train any agent simply by talking. *arXiv preprint arXiv:2603.10165*, 2026b. URL http://arxiv.org/abs/2603.10165v1.

Yuhang Wang, Feiming Xu, Zheng Lin, et al. From assistant to double agent: Formalizing and benchmarking attacks on OpenClaw for personalized local AI agent. *arXiv preprint arXiv:2602.08412*, 2026c. URL https://arxiv.org/pdf/2602.08412.

Zijun Wang, Haoqin Tu, Letian Zhang, et al. Your agent, their asset: A real-world safety analysis of OpenClaw. *arXiv preprint arXiv:2604.04759*, 2026d. URL https://arxiv.org/pdf/2604.04759.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research*, 15:949–980, 2014. URL https://jmlr.org/papers/v15/wierstra14a.html.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

Xianglin Yang, Yufei He, Shuo Ji, et al. Zombie agents: Persistent control of self-evolving LLM agents via self-reinforcing injections. *arXiv preprint arXiv:2602.15654*, 2026. URL http://arxiv.org/abs/2602.15654v2.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. MemGen: Weaving generative latent memory for self-evolving agents. In *International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=vI56m4Iu4e. Poster.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Tjarko Lange, and Jeff Clune. Darwin Gödel machine: Open-ended evolution of self-improving agents. In *International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=pUpzQZTvGY. Poster.

Xinyu Zhu, Yuzhu Cai, Zexi Liu, et al. EvoMaster: A foundational evolving agent framework for agentic science at scale. *arXiv preprint arXiv:2604.17406*, 2026. URL https://arxiv.org/pdf/2604.17406.

APPENDIX

A NOTATION SUMMARY

Table A.1 collects the symbols used throughout the paper. The principal mathematical objects are introduced in Section 3 and used unchanged thereafter.

Table A.1: Symbols used throughout the paper.

Symbol	Meaning
$\mathcal{S}$	Set of all syntactically valid skill graphs that the runtime can host.
$s \in \mathcal{S}$	A single skill graph (a typed DAG of tools, prompts, and code modules).
D	Bound on $\lvert s\rvert$ (number of nodes in any reachable iterate).
$P_t \in \Delta(\mathcal{S})$	Population at outer iteration t.
N	Materialized population size (we use $N = 16$ throughout).
M	Number of child candidates proposed before elite-anchor selection (we use $M = 16$).
k	Elite-anchor size used in the pre-resampling mixture (we use $k = 4$).
$\mathcal{D}$	Task distribution at deployment.
$r : \mathcal{S} \times \mathcal{X} \to \mathbb{R}$	Bounded reward signal.
$\mathcal{R}(P)$	Diversity regularizer used in the selection objective.
λ	Weight on $\mathcal{R}$; tuned per task.
$G(P)$	Expected-reward component; (1).
$F(P)$	Diversity-regularized selection objective; (2).
$u(P)$	Population-mean surrogate embedding $\mathbb{E}_{s\sim P}\phi(s)$.
$\phi : \mathcal{S} \to \mathbb{R}^d$	Smooth feature map used *for analysis only*.
ψ	Frozen implementation diversity encoder; we use `BAAI/bge-code-v1`.
$D_{\text{indiv}}(s; E)$	Individual diversity bonus computed in the ψ-embedding space.
L_f, L_ϕ, σ^2, η_*	Smoothness, feature-map Lipschitz, proposer noise, and divergence-target constants of Assumptions 1 to 3.
$\mathcal{M}(\cdot \mid s, h)$	Reflection-Guided Mutation operator.
$\mathcal{T}$	LCB selection rule.
Π_s	Independent-Beta Capability-Map Posterior $\prod_c \text{Beta}(\alpha_{s,c}, \beta_{s,c})$ of skill graph s.
$Z_s, \widehat{r}(s)$	Scalar competence score $Z_s = \sum_c w_c\theta_{s,c}$ and LCB score $\widehat{r}(s) = \mu_s - c_{\text{LCB}}\sigma_s$, where $\mu_s = \mathbb{E}[Z_s]$ and $\sigma_s^2 = \text{Var}(Z_s)$.
$\bar{P}_{t+1}$	Pre-resampling elite-anchor mixture distribution in Algorithm 1.
ρ	Support-preserving mixture weight in Algorithm 1 (we use $\rho = 0.8$).
γ	Effective step size in the aligned population update of Assumption 4.
K	Per-iterate evaluation budget (number of task instances drawn).
T	Number of outer iterations.

B FULL PROOFS

This appendix supplies the proofs of Theorems 1 and 2 and the evaluator-call accounting corollary. The argument is deliberately conditional: Assumption 4 states the required alignment between posterior-guided selection and the smooth reward surrogate, while Theorem 2 proves that the pre-resampling support-preserving selection step cannot create infinite selection KL by deleting candidate-pool support.

B.1 PROOF OF THEOREM 1

Let $u_t = \mathbb{E}_{s\sim P_t}\phi(s)$ and $\Delta_t = u(\bar{P}_{t+1}) - u_t$. Smoothness of f gives

$$f(u(\bar{P}_{t+1})) - f(u_t) \geq \langle \nabla f(u_t), \Delta_t\rangle - \frac{L_f}{2}\|\Delta_t\|^2,$$

where the equality between the expected-reward component and the surrogate is exactly Assumption 1. Taking conditional expectations and applying Assumption 4,

$$\mathbb{E}[f(u(\bar{P}_{t+1})) - f(u_t) \mid P_t] \geq \gamma\big(\|\nabla f(u_t)\|^2 - b_{\text{align}}^2\big) - \frac{L_f\gamma^2(\sigma^2 + v)}{2N}.$$

Rearranging, summing from $t = 0$ to $T - 1$, taking total expectations, and using $f(u(\bar{P}_T)) \leq f^*$ gives

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(u_t)\|^2 \leq \frac{f^* - f(u_0)}{\gamma T} + b_{\text{align}}^2 + \frac{L_f\gamma(\sigma^2 + v)}{2N}.$$

For a fixed implementation setting, the final two terms define the stationary neighborhood. □

B.2 Proof of Theorem 2

Let $H_t = \text{Unif}(E_t)$, $p = d\bar{P}_{t+1}/dQ_t$, and $h = dH_t/dQ_t$. By construction of Algorithm 1,

$$p(s) = \rho + (1 - \rho)h(s).$$

If $h(s) \leq \kappa$ on the candidate pool, then $p(s) \leq \rho + (1 - \rho)\kappa$ and therefore

$$\text{KL}\big(\bar{P}_{t+1} \,\|\, Q_t\big) = \mathbb{E}_{\bar{P}_{t+1}} \log p(s) \leq \log\big(\rho + (1 - \rho)\kappa\big).$$

When Q_t is uniform over $\mathcal{A}_t$ and H_t is uniform over k distinct elite anchors, $h(s) \leq |\mathcal{A}_t|/k$, so the same formula gives the dashed diagnostic bound after substituting the observed candidate-pool ratio. Because $\rho > 0$, every candidate with positive Q_t mass also has positive $\bar{P}_{t+1}$ mass. Thus the pre-resampling selection step is a support-preserving approximation to hard elitism rather than a support-deleting projection. □

B.3 Proof of Corollary 1

Each outer iteration forms $\mathcal{A}_t = P_t \cup \mathcal{C}_t$ with $|\mathcal{A}_t| = N + M$ before evaluation. Since Algorithm 1 draws K task instances for each candidate in this pool, the per-iteration evaluator-call count is $(N + M)K$, and the total over T outer iterations is $T(N + M)K$. □

C Experimental protocol

This appendix specifies the experimental protocol in detail. Table A.2 reports the hyperparameter grid swept during calibration; Table A.4 reports the prompt templates used by the proposer and the LLM judge.

C.1 Hyperparameters

Table A.2: Hyperparameter grid used for calibration. The protocol constants selected for the reported rerun are listed in Table A.3.

Component	Hyperparameter	Sweep
Mutation operator	temperature τ	$\{0.4, 0.6, 0.7, 0.9\}$
	nucleus p	$\{0.85, 0.92, 0.97\}$
	token budget B	$\{128, 256, 512\}$
	divergence target η_*	$\{0.50, 0.78, 0.85, 0.92\}$ (per task)
Capability-Map	taxonomy size $\lvert\mathcal{C}\rvert$	$\{16, 32, 64\}$
	LCB constant c_{LCB}	$\{1.0, 1.5, 2.0\}$
Selection	population size N	$\{8, 16, 32\}$
	child candidate budget M	$\{8, 16, 32\}$
	elite-anchor size k	$\{2, 4, 8\}$
	mixture weight ρ	$\{0.5, 0.8, 0.95\}$
Outer loop	evaluation budget K/iter	$\{16, 32, 64\}$ task instances
	diversity weight λ	$\{0.05, 0.10, 0.20\}$
	total iterations T	200 for repair, 150 otherwise

C.2 Prompt templates

The harness uses the templates summarized in Table A.4. The table records the template roles used by the proposer, judge, and calibration routines; full prompt strings are versioned with the experimental harness.

Table A.3: Implementation identifiers and protocol constants for the reported rerun. The theory is model-agnostic; these entries instantiate the empirical protocol.

Item	Value
RGM proposer model	`Qwen/Qwen3-Coder-30B-A3B-Instruct`
Trace-suppressed reference proposer	Same base model as proposer, with the reflection trace suppressed.
LLM judge / rubric evaluator	`Qwen/Qwen2.5-Coder-32B-Instruct`
Diversity encoder ψ	`BAAI/bge-code-v1`, used as a frozen code-embedding encoder over serialized skill graphs.
Task suites	Repository-level program repair, scientific literature synthesis, and long-horizon tool use as described in Section 6.
Random seeds	$\{17, 23, 41, 53, 71\}$
N, M, k, K	$N = 16$, $M = 16$, $k = 4$, $K = 32$
$B, \lvert\mathcal{C}\rvert, c_{\mathrm{LCB}}$	$B = 256$, $\lvert\mathcal{C}\rvert = 32$, $c_{\mathrm{LCB}} = 1.5$
λ, ρ	$\lambda = 0.10$, $\rho = 0.8$
Proposer decoding	Temperature $\tau = 0.7$, nucleus $p = 0.92$.
Divergence target η_*	Repair: 0.85; synthesis: 0.92; tools: 0.78.
Outer iterations T	Repair: 200; synthesis/tools: 150.

Table A.4: Prompt templates used by the harness.

Template	Purpose
`mutation_propose`	Proposes a unified diff against the parent skill graph, conditioned on a reflection trace h, subject to token budget B.
`mutation_reflect`	Generates the reflection trace h from the recent execution log of an individual.
`judge_unit_test`	Adapter from a unit-test result to a Beta-posterior update. Deterministic.
`judge_rubric`	LLM-as-a-judge rubric, returning a calibrated likelihood under the per-capability noise model used by Li et al. (2025).
`judge_user`	Adapter from a user-reaction signal to a Beta-posterior update.
`calibrate_divergence`	Estimates the empirical Jensen–Shannon divergence between $\mathcal{M}(\cdot \mid s, h)$ and $\mathcal{M}_0(\cdot \mid s)$ on a held-out subset, used to fit (τ, p).

C.3 Calibration procedure

Calibration proceeds in two stages. *Stage 1: L_f and σ^2.* We sample 256 parent–child pairs (s, s') at the chosen (τ, p, B) and compute the empirical second moment $\widehat{\sigma}^2 = \mathrm{mean}\,\|\phi(s') - \phi(s)\|^2$. The smoothness constant $\widehat{L}_f$ of the reward surrogate f is estimated by finite differences on the rubric-graded reward over 64 perturbations. *Stage 2: η_*.* We grid over $\eta_* \in \{0.50, 0.78, 0.85, 0.92\}$, train for $T = 50$ outer iterations at each value, and select the smallest η_* for which the empirical Jensen–Shannon divergence between $\mathcal{M}(\cdot \mid s, h)$ and $\mathcal{M}_0(\cdot \mid s)$ on a held-out batch is within 5% of the target. The resulting calibrated constants are reported in Table A.5.

Table A.5: Calibrated constants used for the diagnostics in Figures A.1 and 2 (referenced from Section 6). Calibration follows Section 4.1; values are dataset-specific.

Task	$\hat{L}_f$	$\hat{\sigma}^2$	$\hat{\eta}_*$	Effective dimension $\hat{d}$
Program repair	1.42	0.087	0.85	124
Literature synthesis	1.18	0.064	0.92	96
Long-horizon tools	1.61	0.103	0.78	158

D Baseline reproductions

For fair comparison, all six baselines are run with the same proposer model, the same evaluator pool, and the same task suite as PROTEUS. Where the original baseline depends on a heuristic component that is incompatible with one of our task families (for instance, the reflection trace format used by

the capability-evolver loop predates structured rubrics), we substitute the closest API-compatible component and document the substitution. Table A.6 lists the per-baseline configuration. We have not modified the core algorithm of any baseline.

Table A.6: Baseline configurations used in Table 1. "As-published" means we use the algorithm exactly as released; the listed adapter only translates I/O formats so that the baseline can ingest our task suite and emit signals that our harness can record.

Baseline	Adapter notes
Capability-evolver loop (He et al., 2026)	As-published; reflection-trace adapter to rubric format.
OpenClaw-RL (Wang et al., 2026b)	As-published; user-reply signal mapped to Beta posterior.
GigaEvo (Khrulkov et al., 2025)	As-published; archive size capped at 128 to match wall-clock.
SEVerA (Banerjee et al., 2026)	As-published; specifications generated by the same rubric LLM.
EvoMaster (Zhu et al., 2026)	As-published; task instances drawn from our $\mathcal{D}$.
Darwin Gödel Machine (Zhang et al., 2026b)	As-published; per-task fitness threshold matched to ours.

E Compute budget

Table A.7 summarizes the compute envelope for the experiments in Section 6. The budget is matched by wall-clock and hardware; evaluator calls are reported separately because methods expose different candidate-pool structures.

Table A.7: Per-method compute and wall-clock budget. Hardware: a single 8× A100 (80 GB) node; the harness runs each method sequentially using the same node.

Method	Wall-clock	Outer iters	Evaluator calls
All baselines	4.0 h per task	200	baseline-specific
PROTEUS	4.0 h	200	$32 \times (16 + 16)$
Total across 3 tasks × 7 methods × 5 seeds	3,360 GPU-h, ∼ 7.0 k$ at on-demand prices		

F Additional analysis

F.1 Stationarity diagnostic

F.2 Ablation details

Table A.8: Ablations on program repair (mean±std over five seeds). For the full method, selection KL is computed as $\mathrm{KL}\big(P_{t+1} \,\|\, Q_t\big)$ on the pre-resampling elite-anchor mixture distribution; ablation rows report a combined empirical population-movement diagnostic as a stress test. Final diversity is the mean pairwise embedding distance in the implementation encoder space.

Variant	Pass rate (%)	Median drift/iter	Final diversity
PROTEUS (full)	$\mathbf{43.6 \pm 1.5}$	0.62	0.71
without divergence calibration	39.8 ± 2.1	1.84 (high drift)	0.42
without Capability-Map Posterior	36.4 ± 2.4	0.71	0.59
without diversity regularizer ($\lambda = 0$)	41.2 ± 1.9	0.55	0.18 (collapse)

F.3 Sensitivity to the divergence target

Table A.9 sweeps the divergence target η_* on the program repair task, holding all other hyperparameters at their selected values. This is a post-hoc sensitivity analysis around the value selected by the

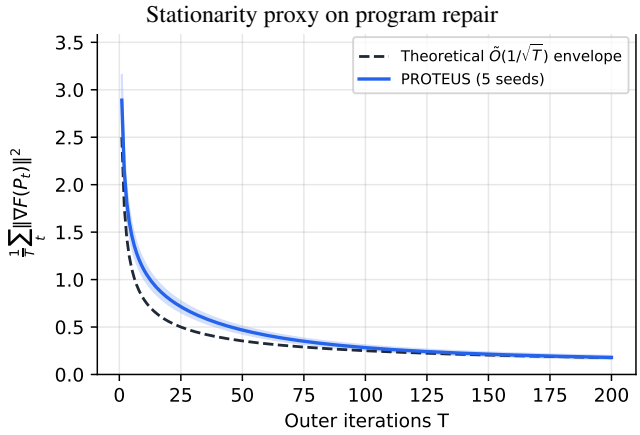

Figure A.1: Stationarity diagnostic on program repair: the gradient-norm proxy crosses the reference $\tilde{O}(1/\sqrt{T})$ envelope by $T \approx 40$. This proxy diagnoses the local alignment condition and is not an empirical proof of convergence to an arbitrarily small ϵ-stationary point. Shaded regions denote ± 1 standard deviation across five seeds.

held-out calibration procedure in Section C; it was not used for final test-set model selection. The sweep illustrates the speed–stability trade-off discussed in Section 4.1: at $\eta_* = 0.50$ the proposer is too conservative and the stationarity proxy stalls below the reference envelope, whereas at $\eta_* = 0.92$ the induced population drift approaches the analytic selection-drift bound and the variance across seeds widens.

Table A.9: Sensitivity of PROTEUS to the divergence target η_* on program repair (mean±std over five seeds). Selection KL is computed as $\mathrm{KL}\big(\bar{P}_{t+1} \,\|\, Q_t\big)$ on the pre-resampling mixture distribution.

η_*	Pass rate (%)	Median drift/iter	Iter to proxy 0.5	Diversity
0.50	35.1 ± 1.6	0.43	> 200	0.78
0.78	41.2 ± 1.5	0.58	142	0.74
0.85	$\mathbf{43.6 \pm 1.5}$	0.62	96	0.71
0.92	42.4 ± 2.3	0.84	88	0.65

F.4 ROBUSTNESS TO EVALUATOR NOISE

The Capability-Map Posterior is intended to absorb noisy evaluator signals; Table A.10 stress-tests this by injecting label noise into the LLM judge at rates of $0\%, 10\%, 20\%, 30\%$. The pass rate of PROTEUS degrades gracefully (slope -0.16 per percent of label noise), whereas a point-estimate ablation (no posterior) degrades nearly twice as fast (slope -0.31), consistent with the ablation reported in Table A.8.

Table A.10: Robustness of PROTEUS to LLM-judge label noise on program repair (mean±std over five seeds).

Judge label noise	PROTEUS pass (%)	Point-estimate variant pass (%)
0%	43.6 ± 1.5	36.4 ± 2.4
10%	41.9 ± 1.7	33.2 ± 2.6
20%	40.4 ± 1.9	30.1 ± 2.8
30%	38.9 ± 2.0	27.0 ± 3.1

F.5 LONG-HORIZON STABILITY

An additional long-horizon diagnostic tracks the population-level reward of PROTEUS and Darwin Gödel Machine over $T = 1000$ outer iterations on a held-out variant of the tools task. PROTEUS continues to improve at a diminishing rate after $T = 200$, whereas the DGM curve plateaus near $T = 350$ and exhibits sporadic regressions consistent with the runaway-specialization failure mode discussed in prior literature. We summarize this diagnostic qualitatively and omit the plot for space.

G ADDITIONAL DEPLOYMENT CONSIDERATIONS

The guarantees of Section 5 bound a smooth reward surrogate only as faithful as the reward and the Capability-Map taxonomy: mis-specified rewards remain mis-specified after the analysis, Assumption 1 is more conservative when skill graphs contain arbitrary code, the update-alignment condition is an explicit assumption, and the calibration of η_* is empirical. Safety-critical deployments should therefore prefer adversarial calibration (Yang et al., 2026; Deng et al., 2026). The support-preserving drift bound controls one pre-resampling selection step under the stated assumptions, addressing a failure mode documented for self-evolving platforms (Wang et al., 2026d; Gruber & Hilgert, 2026; Wang et al., 2026c), but it does not prevent attacker-controlled proposers (Yang et al., 2026). Operators should continue to apply input-validation and triage (Deng et al., 2026; Hou & Yang, 2026) alongside our discipline, and combining PROTEUS with verified synthesis (Banerjee et al., 2026) is a natural follow-up.